Exercise 1

```java
public int remove(int idx) {
    int value = 0;
    if (front == null) {
        throw new IndexOutOfBoundsException();
    }
    if (idx == 0) {
        value = front.data;
        front = front.next;
    } else {
        ListNode current = front;
        int i = 0;
        while (current.next != null && i < idx-1) {
            current = current.next;
            i++;
        }
        if (current.next == null) {
            throw new IndexOutOfBoundsException();
        }
        value = current.next.data;
        current.next = current.next.next;
    }
    size--;
    return value;
}
```

**Diamond 1** → 1 of 2 branches missed  
**Diamond 2** → 1 of 2 branches missed  
**Diamond 3** → 3 of 4 branches missed  
**Diamond 4** → 1 of 2 branches missed  

Exercise 2

```java
public static void add(ListNode list, int data) {
    while (list != null) {
        list = list.next;
    }
    list.next = new ListNode(data);
}
```

(Two overlapping notifications)

1 - Null pointer dereference of list in  
`edu.ncsu.csc216.linked_list.ListNode.add(ListNode, int)`  

A null pointer is dereferenced here. This will lead to a NullPointerException when the code is executed.
2 - Load of known null value in edu.ncsu.csc216.linked_list.ListNode.add(ListNode, int)

The variable referenced at this point is known to be null due to an earlier check against null. Although this is valid, it might be a mistake (perhaps you intended to refer to a different variable, or perhaps the earlier check to see if the variable is null should have been a check to see if it was non null).

Exercise 3

```java
public Object next() {
    Object result = current.data;
    current = current.next;
    return result;
}
```

linked_List_Iterator_Bug.LinkedIterator.next() can’t throw NoSuchElementException

This class implements the java.util.Iterator interface. However, its `next()` method is not capable of throwing java.util.NoSuchElementException. The `next()` method should be changed so it throws NoSuchElementException if is called when there are no more elements to return.

Exercise 4

```java
public static int pow1(int base, int exp) {
    return (base * pow1(base, exp-1));
}

public static int pow2 (int n, int m) {
    if (m == 1) {
        return n;
    } else {
        return (n * pow2(n, m));
    }
}
```

Bug: There is an apparent infinite recursive loop in infinite_Recursion_Bug.Recursive_Methods.pow1(int, int)

This method unconditionally invokes itself. This would seem to indicate an infinite recursive loop that will result in a stack overflow.