Challenge: Frauds List

Here is a video walkthrough of the solutions.

(6 Points). Suppose we have the `IntList` and `FraudsList` classes below (Summer 2021, Final)

```
public class IntList {
    public int first;
    public IntList rest;

    public IntList(int f, IntList r) {
        first = f;
        rest = r;
    }

    public int size() {
        IntList p = this;
        int totalSize = 0;
        while (p != null) {
            totalSize += 1;
            p = p.rest;
        }
        return totalSize;
    }
}

class FraudList extends IntList {
    public FraudList(int f, IntList r) {
        super(f, r);
    }
    public int size() {
        return -super.size();
    }
}
```

Implement the method `findFrauds` which accepts an array of `IntLists` in which some of the elements are, or may contain, `FraudLists`! That is, the dynamic type of certain `IntList` instances is `FraudList`. As shown above, a `FraudList` is an `IntList` whose size method returns the negative of the correct size. You must report these `FraudLists` by **non-destructively** returning a **new** `FraudList` of all the `FraudList` instances linked together in the order they appear in `arr`.

You may **not** modify the given array `arr` or the `IntLists` inside of `FraudList`. You may **not** use `instanceOf`, `getClass()`, `isInstance()` or any method not explicitly written in the classes above or imported. An instance of the problem is shown below:

```
IntList first = new IntList(1000, new IntList(1002, new FraudList(1, new FraudList(2, null))));
IntList second = new FraudList(3, null);
```

```
IntList third = new IntList(3000, null);
IntList fourth = new FraudList(4, new IntList(231, new FraudList(5, null)));
IntList[] arr = new IntList[]{first, second, third, fourth};
FraudList frauds = findFrauds(arr);
```

After executing the lines above, `frauds` should be equal to the `FraudList` with the elements `1, 2, 3, 4, 5` and **`arr`, as well as the contents within `arr`, should be unchanged**. Fill in the skeleton below. You may not delete, modify, or add to any of the provided skeleton code.

```
import static java.lang.System.arraycopy;

public static FraudList findFrauds(IntList[] arr) {
    IntList[] copy = new IntList[arr.length];
    arraycopy(arr, 0, copy, 0, arr.length);
    return helper(____________, ____________);
}

public static FraudList helper(IntList[] copy, int index) {
    if (___________________________________) {
        return null;
    } else if (_____________________________) {
        return ______________________________;
    }
    ____________________________________________;
    ____________________________________________;
    if (___________________________________) {
        return ______________________________;
    } else {
        return ______________________________;
    }
}
```

Solution:

```
import static java.lang.System.arraycopy;

public static FraudList findFrauds(IntList[] arr) {
    IntList[] copy = new IntList[arr.length];
    arraycopy(arr, 0, copy, 0, arr.length);
    return helper(copy, 0);
}

public static FraudList helper(IntList[] copy, int index) {
    if (index == copy.length) {
        return null;
    } else if (copy[index] == null) {
        return helper(copy, index + 1);
```

```
14      }
15      IntList current = copy[index];
16      copy[index] = current.rest;
17      if (current.size() < 0) {
18          return new FraudList(current.first, helper(copy, index));
19      } else {
20          return helper(copy, index);
21      }
22  }
```