Pokud zadání nerozumíte nebo se vám zdá nejednoznačné, zeptejte se. Pište čitelně, nečitelná řešení nebudeme uznávat.

1. Odkrokujte metodu `N2.main` a s použitím notace z přednášky popište stav paměti v místech označených komentáři (stavy A, B a C). Pokud bude daný řádek programu navštíven vícekrát, vypište stav pro každou návštěvu.

```
interface Node {
    boolean contains(int value);
}

class N1 implements Node {
    final int val;
    final Node next;
    static int count = 0;

    public N1(int val, Node next) {
        this.val = val;
        this.next = next;
        count++;
        // stav A
    }

    @Override
    public boolean contains(int value) {
        if (val == value) {
            // stav B
            return true;
        }
        else {
            // stav C
            return next.contains(value);
        }
    }
}

class N2 implements Node {
    @Override
    public boolean contains(int value) {
        return false;
    }

    public static void main(String[] args) {
        Node n = new N1(1, new N1(2, new N1(3, new N2())));
        n.contains(2);
        n.contains(4);
    }
}
```

2. Popište, co v obecném případě vrací metody `f` a `g`, a co vrátí volání `new CC(8).g(9)`.

```
class CC {
    final int x;

    public CC(int x) {
        this.x = x;
    }

    int f(int y) {
        return x == 0 ? y : new CC(x - 1).f(y + 1);
    }

    int g(int y) {
        return x == 1 ? y : new CC(new CC(x - 1).g(y)).f(y);
    }
}
```

3. Najděte a vysvětlete chybu **při překladu**.

```
class Set {
    void clear() {
        System.out.println("Set.clear()");
    }
}

class List extends Set {
    void empty() {
        System.out.println("List.empty()");
    }
}

class Task3 {
    public static void main(String[] args) {
        Object a = new Set();
        Object b = new List();
        if (!a.equals(b)) {
            List c = (List) a;
            c.empty();
            c.clear();
            Set d = (List) b;
            d.empty();
            c.clear();
        }
    }
}
```

4. Vyznačte řádek, na kterém dojde k chybě za běhu programu. K jakému typu chyby dojde (nemusíte psát přesný název výjimky, stačí popsat typ chyby, např. chybné přetypování, přetečení zásobníku apod.)? Dojde k chybě během volání `contains(2)` nebo `contains(4)`? Proč?

```
class N1 implements Node {
    final int val;
    final Node next;

    public N1(int val, Node next) {
        this.val = val;
        this.next = next;
    }

    public boolean contains(int value) {
        if (val == value)
            return true;
        else
            return ((N1) next).contains(value);
    }
}

class N2 implements Node {
    public boolean contains(int value) {
        return false;
    }

    public static void main(String[] args) {
        N1 n = new N1(1, new N1(2, new N1(3, new N2())));
        n.contains(2);
        n.contains(4);
    }
}
```

5. Třída `MultiSet` představuje multimnožinu, tj. množinu, v níž se mohou prvky opakovat. Dopište kód metody `void remove(Object element)`, která sníží četnost opakování prvku `element` v multimnožině o 1.

```
class MultiSet {
    private Object[] elements = new Object[100000];
    private int size = 0;

    public int getCount(Object element) {
        int result = 0;
        for (int i = 0; i < size; i++)
            if (elements[i].equals(element)) result++;
        return result;
    }

    public void add(Object element) {
        elements[size++] = element;
    }

    public void remove(Object element) {









    }
}
```