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

1. Odkrokujte následující program a s použitím notace z přednášky popište stav paměti v místech označených komentáři (stav A, stav B). Pokud bude daný řádek programu navštíven vícekrát, vypište stav pro každou návštěvu. Předpokládejte, že garbage collector odstraní z haldy alokované instance okamžitě poté, co přestanou být z programu dostupné.

```
public class Task1 {
    private static int COUNTER = 0;

    int f(int n) {
        COUNTER++;
        if (n >= 0) {
            int eo = n % 2 == 0 ? e(n) : o(n);
            Task1 t = new Task1();
            //stav A
            return t.f(n - 1) + eo;
        }
        return 0;
    }

    int o(int n) {
        return 2 * n;
    }

    int e(int n) {
        return 2 * n + 1;
    }

    public static void main(String[] args) {
        Task1 t = new Task1();
        int v = t.f(2);
        //stav B
        // ...
    }
}
```

Následující seznam ukazuje stav paměti, na místech označených komentáři.

stav A:

$$\left(\begin{array}{l|l|l} \#1 \to Task1() & this = \#1, n = 2, eo = 5, t = \#2 & Task1.COUNTER = 1 \\ \#2 \to Task1() & \overline{\qquad\qquad\qquad} & \\ & t = \#1 & \end{array}\right)$$

stav A:

$$\left(\begin{array}{l|l|l} \#1 \to Task1() & this = \#2, n = 1, eo = 2, t = \#3 & Task1.COUNTER = 2 \\ \#2 \to Task1() & \overline{\qquad\qquad\qquad} & \\ \#3 \to Task1() & this = \#1, n = 2, eo = 5, t = \#2 & \\ & \overline{\qquad\qquad\qquad} & \\ & t = \#1 & \end{array}\right)$$

stav A:

$$\left(\begin{array}{l|l|l} \#1 \to Task1() & this = \#3, n = 0, eo = 1, t = \#4 & Task1.COUNTER = 3 \\ \#2 \to Task1() & \overline{\qquad\qquad\qquad} & \\ \#3 \to Task1() & this = \#2, n = 1, eo = 2, t = \#3 & \\ \#4 \to Task1() & \overline{\qquad\qquad\qquad} & \\ & this = \#1, n = 2, eo = 5, t = \#2 & \\ & \overline{\qquad\qquad\qquad} & \\ & t = \#1 & \end{array}\right)$$

stav B:

$$\left(\ \#1 \to Task1() \ \middle|\ t = \#1, v = 8 \ \middle|\ Task1.COUNTER = 4\ \right)$$

2. Napište, co vytiskne následující kód.

```
abstract class I {
    final int a;
    final int b;
    final I i;

    I(int a, int b, I i) {
        this.a = a;
        this.b = b;
        this.i = i;
    }

    abstract int f();
}

class F extends I {
    F() { super(10, 0, null); }

    public int f() { return a; }
}

class S extends I {
    S(int a, int b, I i) { super(a, b, i); }

    public int f() { return a * i.f() + b * i.f(); }
}

class M extends I {
    M(int a, int b, I i) { super(a, b, i); }

    public int f() { return (a + i.f()) * (b + i.f()); }
}

public class Task2 {
    public static void main(String[] args) {
        System.out.println(new M(1, 2, new S(1, 2, new F())).f());
        System.out.println(new S(1, 2, new M(1, 2, new F())).f());
    }
}
```

Kód vypíše na samostatné řádky čísla: 992, 396.

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

```
class A {
    void f() {
        System.out.println("A.f()");
    }
}

class B extends A {
    void g() {
        System.out.println("B.g()");
    }
}

public class Task3 {
    public static void main(String[] args) {
        Object a = new A();
        Object b = new B();
        if (!a.equals(b)) {
            A c = (B) a;
            c.f();
            A d = (B) b;
            d.g();
        }
    }
}
```

Chyba při překladu nastane pro řádek: `d.g()` metody `main()`: metoda `g()` je definována až ve třídě `B`. Matoucí může být řádek `A c = (B) a`, na tom by ale došlo až k běhové chybě (snažíme se přetypovat instanci předka `A` na typ potomka `B`, tj. `ClassCastException`).

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.)? Vytiskne program text „LOPATA“?

```
interface T {
    int f(int v);
}

class X implements T {
    private int[] a;

    X() { this.a = new int[]{1, 3, 2, 5, 6, 0, 4}; }

    public int f(int v) { return a[v]; }
}

class Y implements T {
    private int i;

    Y(int i) { this.i = i; }

    public int f(int v) { return new X().f(i + v); }
}

class Z extends Y {

    Z() { super(4); }

    public int f(int v) { return super.f(-v); }
}

public class Task4 {
    public static void main(String[] args) {
        System.out.println(new Z().f(-2));
        System.out.println("LOPATA");
        System.out.println(new Z().f(5));
    }
}
```

Text „LOPATA“ je vypsán. Problém nastane až při volání `new Z().f(5)`. Následuje chronologický výpis stavů paměti bezprostředně po každém zavolání (libovolné implemetace) metody `f(int v)`. Pole a ve třídě `X` je ve výpisech ignorováno.

`Z.f(5)`:

$$\left(\begin{array}{l|l} \#3 \to Z(i=4) & \begin{array}{l} this = \#3, v = 5 \\ \hline \end{array} \end{array} \right)$$

`Y.f(-5)`:

$$\left(\begin{array}{l|l} \#3 \to Z(i=4) & this = \#3, v = -5 \\ & \hline & this = \#3, v = 5 \\ & \hline \end{array} \right)$$

`X.f(-1)`:

$$\left(\begin{array}{l|l} \#3 \to Z(i=4) & this = \#4, v = -1 \\ \#4 \to X(a=NA) & \hline & this = \#3, v = -5 \\ & \hline & this = \#3, v = 5 \\ & \hline \end{array} \right)$$

Protože metoda `X.f(int v)` používá svůj parametr pro přístup do pole `a`. Dojde k chybě: `ArrayIndexOutOfBoundsException: -1`.

5. Třída `MultiSet` představuje multimnožinu, tj. množinu, v níž se mohou prvky opakovat. Dopište kód metody `void add(Object element)`, která do multimnožiny vloží nový prvek.

```
class MultiSet {
    private Map<Object, Integer> counts = null;

    public int getCount(Object element) {
        if (counts != null && counts.containsKey(element)) {
            return counts.get(element);
        }
        return 0;
    }

    public void add(Object element) {
        if (counts == null) { // lazy initialization
            counts = new HashMap<>();
        }
        Integer count = counts.get(element);
        counts.put(element, count == null ? 1 : count + 1);
    }

    public static void main(String[] args) {
        MultiSet m = new MultiSet();
        m.add("Ahoj");
        m.add("Cau");
        m.add("Ahoj");
        System.out.println(m.getCount("Ahoj"));  // tiskne 2
        System.out.println(m.getCount("Cau"));   // tiskne 1
        System.out.println(m.getCount("Nazdar"));// tiskne 0
    }
}
```