

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

1. Odkrojujte 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
        // ...
    }
}
```

OMO, 17. 5. 2013

Jméno:

---

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());
    }
}
```

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();
        }
    }
}
```

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));
    }
}
```

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) {

    }
}
```