A. Co vypíše metoda `f`? Proč?

```
class Cell {
    int contents;

    Cell(int c) { contents = c; }
    void set(int c) { contents = c; }
    int get() { return contents; }

    static void f() {
        Cell c = new Cell(4);
        g(c);
        c.set(c.get() + 1);
        System.out.println(c.get());
    }

    static void g(Cell c) {
        c.set(c.get() + 1);
        c = new Cell(8);
    }
}
```

B. Instance třídy `Set` reprezentují množinu – metoda `add` vloží prvek, metoda `remove` odstraní prvek a metoda `contains` vrátí `true` nebo `false` podle toho, jestli je daný prvek v množině nebo ne. Třída `Ticket` ji využívá pro evidenci platných čísel – platné číslo je takové, které bylo přidáno metodou `addValidNumber` a nebylo zneplatněno metodou `invalidateNumber`. Co je na implementaci třídy `Ticket` špatně?

```
class Set {
    /* atributy */
    void add(int element)          { /* kod */ }
    void remove(int element)       { /* kod */ }
    boolean contains(int element) { /* kod */ }
}

class Ticket {
    Set validNumbers, invalidNumbers;

    Ticket() {
        validNumbers = invalidNumbers = new Set();
    }

    void addValidNumber(int element) {
        validNumbers.add(element);
    }

    void invalidateNumber(int element) {
        invalidNumbers.add(element);
    }

    boolean isValidNumber(int element) {
        return validNumbers.contains(element) &&
            ! invalidNumbers.contains(element);
    }
}
```

C. Předpokládejte, že třída `Stack` reprezentuje zásobník – metoda `push` vkládá číslo do zásobníku, metoda `pop` vrací nejmladší vložené číslo (a odstraní ho ze zásobníku) a metoda `isEmpty` vrací `true` nebo `false` podle toho, jestli je zásobník prázdný. Naimplementujte třídu `PriorityStack`, kde metoda `push` dostane ještě parametr `true` nebo `false` podle toho, jestli má vkládaný prvek vysokou prioritu – prvky s vysokou prioritou jsou pokládány za mladší než prvky s nízkou prioritou bez ohledu na to, kdy byly reálně vloženy. Vyhněte se duplikaci kódu.

```
class Stack {
    /* atributy */

    void push(int elem) { /* kod */ }
    int pop()           { /* kod */ }
    boolean isEmpty()   { /* kod */ }
}

class PriorityStack {
    /* atributy */

    void push(int e, boolean highPriority) { /* kod */ }
    int pop()         { /* kod */ }
    boolean isEmpty() { /* kod */ }
}
```

Objektové modelování – zpětná vazba uprostřed semestru

Cílem předmětu Objektové modelování je naučit vás objektově programovat a především objektově myslet. Aby se nám to co nejlépe podařilo, potřebovali bychom vědět vaše odpovědi na čtyři níže uvedené otázky.

Předmět Algoritmizace vás na OMO připravil

☐ nedostatečně

☐ dostatečně

☐ byl zbytečně podrobný

Přednáška vám k pochopení látky pomáhá

☐ nedostatečně

☐ dostatečně

☐ (většinou) nechodím

Cvičení vám k pochopení látky pomáhá

☐ nedostatečně

☐ dostatečně

☐ (většinou) nechodím

Učebnice vám k pochopení látky pomáhá

☐ nedostatečně

☐ dostatečně

☐ nemám

Komentář

Pokud máte něco na srdci, nebojte se to napsat.