Zadanie: Vytvorte triedu, ktora bude vytvarat objekty reprezentujuce logicke vyrazy. Logicky vyraz (v tomto konkretnom pripade) bude pozostavat z n premennych a logickych operacii: & - pre logicky sucin, | - pre logicky sucet a ^ - pre logicku negaciu.
Trieda bude dalej obsahovat (najmenej) tieto metody (s lubovolnymi parametrami):

1. ohodnotVyraz() - metoda pre zadane vstupne hodnoty premennych urci konecnu hodnotu logickeho vyrazu (True, False).
2. zistiKedyJeTrue() - metoda, ktora bude zistovat pre ktore hodnoty vstupnych premennych je hodnota logickeho vyrazu True.

Priklad:

```
public static void main(String[] args) {
    LogickyVyraz logVyraz = new LogickyVyraz("(((X&Y)&(X|Y))|(^Z))");
    System.out.println(logVyraz);

    logVyraz.zadajVstupnePremenne('X', '1');
    logVyraz.zadajVstupnePremenne('Y', '1');
    logVyraz.zadajVstupnePremenne('Z', '0');

    System.out.println("Hodnota log. vyrazu pre hodnoty "+logVyraz.hodnoty+": "
    + logVyraz.ohodnotVyraz(logVyraz) + ".( 1=TRUE ; 0=FALSE ; e=chyba )");

    logVyraz.zistiKedyJeTrue('X');
}
```

Zopar hintov:

- Na metodu ohodnotVyraz() som nasiel niekolko moznych rieseni, ale iba jedno som implementoval (samozrejme to najjednoduchsie). Da sa to riesit rekurzivne nasledujucim sposobom: Vyraz sa da rozdelit na niekolko podproblemov z ktorych uplne najspodnejsie budu samotne vyrazy A&B, A&C a ^A a pod.. Postupne tak mozeme urcovat ich hodnotu a vynarat sa z jednotlivych urovni.
  Napr.:

  ```
  ((1&0)|(1&1))|(^1))
  ((0|1)|(0|0))
  (1|0)
  1
  ```

  Ako mozete vidiet pri ^1 som si vysledok previedol do tvaru binarnej operacie 0|0 - toto je (ako mozno zistite pri rieseni) uzitocne.

- Dalsia metoda je v podstate podobna tej prvej, ale nepouziva rekurziu, ale zasobnik. To znamena ze pri prvom behu upravi "zakladne vyrazy" typu A&B, ^A a pod. a vysledne hodnoty postupne uklada do zasobnika, spolu so spravnym uzatvorkovanim. Cize napr. pri prvom behu upravi vyraz ((1&0)|(1&1))|(^1)) na (1|1)|(0|0). Potom jednoducho vyberie vsetky hodnoty do zasobnika, ulozi ich do pola ktore sa spracovava. Cyklus skonci ked je v poli len jedna hodnota, a to vysledok.

- Nezabudnite v rekurzii je rozdiel medzi post-increment (i++) a pre-increment (++i). Pretoze ked date volat rekurzivne napr. robNieco(i++) tak preposle hodnotu i a nie i zvacsenu hodnotu o jedna, pretoze post-increment najprv priradi premennu a az potom ju zvacsi - to je jeden taky maly problemik, na ktory mozno narazit.