Zadanie programistyczne 2

Napisz program, wykonujący analizę zasięgu zmiennych w podanym kodzie w języku Java.

Wprowadzenie: zasięg zmiennej

W Javie, podobnie jak w wielu innych imperatywnych językach programowania, kluczową rolę pełnią **zmienne**, w których tymczasowo przechowujemy różne wartości. Każdą zmiennę trzeba przed użyciem **zadeklarować**, na przykład pisząc

```
int i;       // deklaracja bez inicjalizacji
```

lub

```
int i = 23;  // deklaracja z inicjalizacją
```

Po zadeklarowaniu zmiennej można używać, jednak nie w całym programie, lecz jedynie w pewnym obszarze, zwanym **zasięgiem** tej zmiennej. Zasięg rozpoczyna się od deklaracji zmiennej, zaś kończy się wraz z końcem najmniejszego **bloku** kodu (obszaru ograniczonego parą nawiasów `{` oraz `}`) zawierającego tę deklarację (z wyjątkiem parametrów funkcji – patrz niżej).

Na przykład, w kodzie

```
1  int fun1(int c) {
2    int a = 2;
3    if (a > 0) {
4      int b = 3;
5      b = b * (a + c);
6      return b;
7    }
8  }
```

zasięg zmiennej `a` rozpoczyna się od jej deklaracji w linii 2, a kończy na nawiasie `}` w linii 8, natomiast zasięg zmiennej `b` rozpoczyna się od linii 4, a kończy na wewnętrznym nawiasie `}` w linii 7.

Oprócz tego mamy jeszcze symbol `c`, zadeklarowany w linii 1 jako **parametr** funkcji – takie przypadki w tym zadaniu również potraktujemy jak zmienne (w tym przypadku deklaracja `c` znajduje się w linii 1). Jedyną różnicą jest to, że zasięg `c` ograniczony jest do bloku będącego ciałem funkcji, a więc następującego zaraz po deklaracji `c`. Zatem zasięg c rozpoczyna się w linii 1, a kończy w linii 8 (podobnie jak dla `a`).

Umiejętność ustalenia zasięgu zmiennych jest ważną cechą kompilatora. Pozwala ona m.in. na sprawdzenie już podczas kompilacji, czy użycie zmiennych przez programistę jest poprawne. (A czasem również na optymalizację kodu wynikowego – patrz część bonusowa).

Wejście

Wejściem dla programu jest zawartość pojedynczego pliku z kodem Javy, zawierającego definicję jednej klasy, wyposażonej w pewną liczbę metod.

Dla uproszczenia proszę założyć, że:

- podany kod stanowi poprawną zawartość pliku z kodem Javy
- wszystkie zmienne są typów prostych (`boolean` / `byte` / `char` / `short` / `int` / `long` / `float` / `double`)
- nazwy wszystkich zmiennych są jednoliterowe
- wszystkie deklarowane zmienne (ale nie parametry funkcji) są natychmiast inicjowane (np. `int i = 0;` , a nie samo `int i;`)
- program nie zawiera:
  - jednoliterowych identyfikatorów (ani słów kluczowych) innych niż nazwy zmiennych
  - instrukcji `import`
  - definicji jakichkolwiek innych klas
  - deklaracji atrybutów (pól) klasy
  - komentarzy
  - literałów typu `char` (np. `'a'`) ani `String` (np. `"abc"`)
  - instrukcji `for` oraz `switch`
  - znaków spoza zestawu ASCII
  - operatorów przypisania innych niż = (a więc brak `++`, `--`, `+=` itp.)

Z drugiej strony, proszę **nie zakładać** nic na temat formatowania - podział na linie oraz wcięcia mogą być dowolnie nietypowe (na przykład cały program w jednej linii, albo każdy identyfikator w osobnej linii, itd.)

Jeśli odczuliby Państwo potrzebę dodatkowych założeń na temat wejścia, proszę o kontakt.

Wyjście

Dla każdej deklaracji zmiennej w kodzie wejściowym, program powinien wypisać jedną linię postaci:

zm dekl_wiersz dekl_kol koniec_zas_wiersz koniec_zas_kol

gdzie:

- *zm* to nazwa zmiennej (zawsze 1 litera);
- *dekl_wiersz*, *dekl_kol* to numer wiersza i kolumny w kodzie, gdzie występuje deklaracja zmiennej (a konkretnie – nazwa zmiennej w ramach jej deklaracji);
- *koniec_zas_wiersz*, *koniec_zas_kol* to numer wiersza i kolumny w kodzie, gdzie występuje nawias `}` kończący zasięg tej zmiennej.

Uwaga: wiersze i kolumny w kodzie tradycyjnie numerujemy począwszy od 1.

Powyższe linie należy wypisać na wyjście w kolejności zgodnej z kolejnością występowania deklaracji w kodzie wejściowym.

Przykłady

Przykład 1

Dla wejścia

```
public class MyClass {
  public void printThree() { int i = 3; System.out.println(i); }
  void printFirstTenSquares() {
```

```
    int i = 0;
    while (i <= 10) {
      int y = 3;
      printNumber(i * i, 0); i = i + 1;
    }
  }
  void printNumber(int x, int y) {
    System.out.println(x);
  }
}
```

program powinien wypisać:

```
i 2 34 2 64
i 4 9 9 3
y 6 11 8 5
x 10 24 12 3
y 10 31 12 3
```

Przykład 2

Dla wejścia

```
public class MyClass {
  public static void doSomething(int a, int b) {
    int c = b; int e = b; int d = 6;
    if (d > 3) { if (c > 3) {
      int c = 4; int d = 5;
      System.out.println(c + e); } System.out.println(d); }
  }
}
```

program powinien wypisać:

```
a 2 38 7 3
b 2 45 7 3
c 3 9 7 3
e 3 20 7 3
d 3 31 7 3
c 5 11 6 34
d 5 22 6 34
```

Bonus (20% punktów)

Dodatkowe **20% punktów** można otrzymać za wzbogacenie programu o **analizę żywotności** zmiennych. O ile zasięg zmiennej jest obszarem kodu, gdzie użycie danej zmiennej jest dozwolone (zatem rozciąga się aż do końca odpowiedniego bloku), o tyle **obszar żywotności** zmiennej kończy się wcześniej – w momencie ostatniego użycia wartości tej zmiennej.

Pojęcie to jest o tyle ważne dla kompilatora, że wygenerowany przezeń kod może nie przechowywać wartości zmiennych poza obszarem ich żywotności, co niekiedy pozwala na "zmieszczenie" większej liczby zmiennych w mniejszej liczbie rejestrów. Najprostszy przykład takiej sytuacji mógłby wyglądać tak:

```
void func() {
  int a = 1;
  System.out.println(a);
  int b = 2;
  System.out.println(b);
}
```

Obszarem żywotności zmiennej `a` są linie 2 i 3, zaś zmiennej `b` – linie 4 i 5. Ponieważ te obszary są rozłączne, kompilator może z łatwością zaalokować obie te zmienne do tego samego rejestru!

Specyfikacja bonusu

Bonus polega na modyfikacji rozwiązania podstawowego tak, by wczytywało nadal wejście o tym samym formacie i założeniach, jednak każda linia wyjścia ma być rozbudowana do postaci

zm dekl_wiersz dekl_kol koniec_zas_wiersz koniec_zas_kol koniec_zyw_wiersz koniec_zyw_kol

gdzie:

- pierwszych 5 członów ma znaczenie jak dotychczas,
- *koniec_zyw_wiersz*, *koniec_zyw_kol* to:
  - numer wiersza i kolumny w kodzie, gdzie występuje ostatnie użycie wartości danej zmiennej;
  - dwa znaki `-`, jeśli wartość danej zmiennej nie jest nigdy użyta.

Przykład dla bonusu

Dla wejścia z Przykładu 2, program bonusowy powinien wypisać:

```
a 2 38 7 3 - -
b 2 45 7 3 3 24
c 3 9 7 3 4 22
e 3 20 7 3 6 30
d 3 31 7 3 6 55
c 5 11 6 34  6 26
d 5 22 6 34 - -
```

Uwaga: w przypadku kolizji nazw zmiennych, trzeba tu mieć na uwadze **przesłanianie** zmiennych "zewnętrznych" przez zmienne "wewnętrzne" o tej samej nazwie, i prześledzić, która z nich jest używana w którym miejscu. W tym przypadku stwierdzamy, że:

- "Zewnętrzna" zmienna `c` (zadeklarowana w linii 3) ulega przesłonięciu przez "wewnętrzną" zmienną `c` (zadeklarowaną w linii 5), wobec czego odwołanie do zmiennej `c` w linii 5 odnosi się do zmiennej "wewnętrznej". Z tego względu obszar żywotności "zewnętrznego" `c` kończy się wcześnie, już w linii 4.
- Podobnie, "zewnętrzna" zmienna `d` (zadeklarowana w linii 3) ulega przesłonięciu przez "wewnętrzną" zmienną `d` (zadeklarowaną w linii 5), jednak odwołanie w linii 6 znajduje się już poza zasięgiem zmiennej "wewnętrznej", zatem musi się odnosić do zmiennej "zewnętrznej". Wobec tego zmienna "wewnętrzna" jest nieużywana, zaś obszar żywotności "zewnętrznej" kończy się dopiero w linii 6.

Wskazówka

Nie warto zbyt głęboko próbować "zrozumieć" kodu Javy podanego na wejściu - byłoby to dość trudne, i niepotrzebne w tym zadaniu. W wersji podstawowej wystarczy się skupić na zidentyfikowaniu:

- gdzie znajdują się deklaracje zmiennych, a gdzie ich użycia?
  - tu pomoże Państwu założenie, że nazwy zmiennych są jednoliterowe
- gdzie zaczynają się i kończą poszczególne metody?
  - w tym celu wystarczy zliczać nawiasy klamrowe `{` oraz `}`
    (co jest prostsze dzięki założeniu, że program nie zawiera komentarzy ani literałów tekstowych)

W wersji bonusowej trzeba zidentyfikować nieco więcej elementów programu, jednak wciąż nie ma potrzeby głębokiego parsowania.

Inne uwagi

Tak jak w zadaniu 1, nie ograniczam wyboru języka, w którym napiszą Państwo analizator. (Natomiast wejście dla niego jest kodem w Javie).

Przesyłanie rozwiązań

Rozwiązania proszę umieszczać **do 29 maja** w systemie Edux.