Laboratorium 3
Cel: budowa własnych klas i stosowanie obiektów w programach

1. Rozwiązywanie równania kwadratowego – definiowanie własnej klasy *Rownanie*
   1.1. Działanie programu: Wykonaj program, który oblicza wartość równia kwadratowego $ax^2+bx+c=0$ dla zadanych wartości parametrów a, b, c:
   - dla x=0 brak równania kwadratowego
   - delta jest dana wzorem d=b2-4ac
   - dla d<0 nie ma pierwiastków rzeczywistych
   - dla d=0 jest podwójny pierwiastek równy x1=x2=-b/(2a)
   - dla d>0 mamy dwa pierwiastki dane wzorami
     x1=(-b-sqrt(d))/(2a)
     x2=(-b+sqrt(d))/(2a)

   1.2. Algorytm programu:
   a) należy z klawiatury podać wartości a i sprawdzić, czy a<>0 - jeśli nie, wyświetlić komunikat i zakończyć program
   b) należy z klawiatury podać wartości b i c
   c) obliczyć wartość delty d=b*b-4*a*c
   d) sprawdzić, czy delta jest nieujemna, d>=0- jeśli nie, wyświetlić komunikat i zakończyć program
   e) obliczyć pierwiastek z delty d= sqrt(d)
   f) obliczyć a=2*a
   g) sprawdzić, czy delta d ==0- jeśli tak, obliczyć podwójny pierwiastek x1= -b/a, wyświetlić jego wartość na ekranie i zakończyć program
   h) obliczyć dwa pierwiastki:
   x1= (-b-d)/a
   x2=(-b+d)/a,
   wyświetlić ich wartości na ekranie i zakończyć program

   1.3. zdefiniowanie klasy *Rownanie*, która zawiera:
   a) atrybuty statyczne typu parametry *a, b, c* równania oraz atrybuty *x1* oraz *x2* reprezentujące rozwiązanie tego równania
   b) następujące metody*: czy_rownanie_kwadratowe, oblicz_delta, czy_jest_rozwiazanie, oblicz_pierwiastki*

   1.4. Napisz program, który używa metod klasy *Rownanie* do obliczenia pierwiastków. Dane należy wprowadzać z klawiatury za pomocą *JOptionPane.showInputDialog* oraz prezentować za pomocą metod: *JOptionPane.showMessageDialog* oraz *System.out.println*

****

Algorytmy z pętlami

2. Ogólne sformułowanie algorytmu sortowania bąbelkowego

2.1. Algorytm sortowania N elementów (koncepcja):

(1) *wykonaj, co następuje, N -1 razy*
(1.1) *wskaż na pierwszy element;*
(1.2) *wykonaj, co następuje, N - 1 razy:*
(1.2.1) *porównaj wskazany element z elementem następnym*
(1.2.2) *jeśli porównywane elementy są w niewłaściwej kolejności, zamień je miejscami;*
(1.2.3) *wskaż na następny element.*

2.2. Algorytm sortowania N elementów(projekt):

(1) *i ← 1;*
(2) *dopóki i <= N -1, wykonuj co następuje:*
(2.1) *j ← 1;*
(2.2) *dopóki j <= N - i, wykonuj, co następuje:*
(2.2.1) *jeśli T(j + 1) < T(j), to zamień je;*
(2.2.2) *j ← j + 1;*
(2.3) *i ← i +1;*

2.2. Algorytm programu następujący:

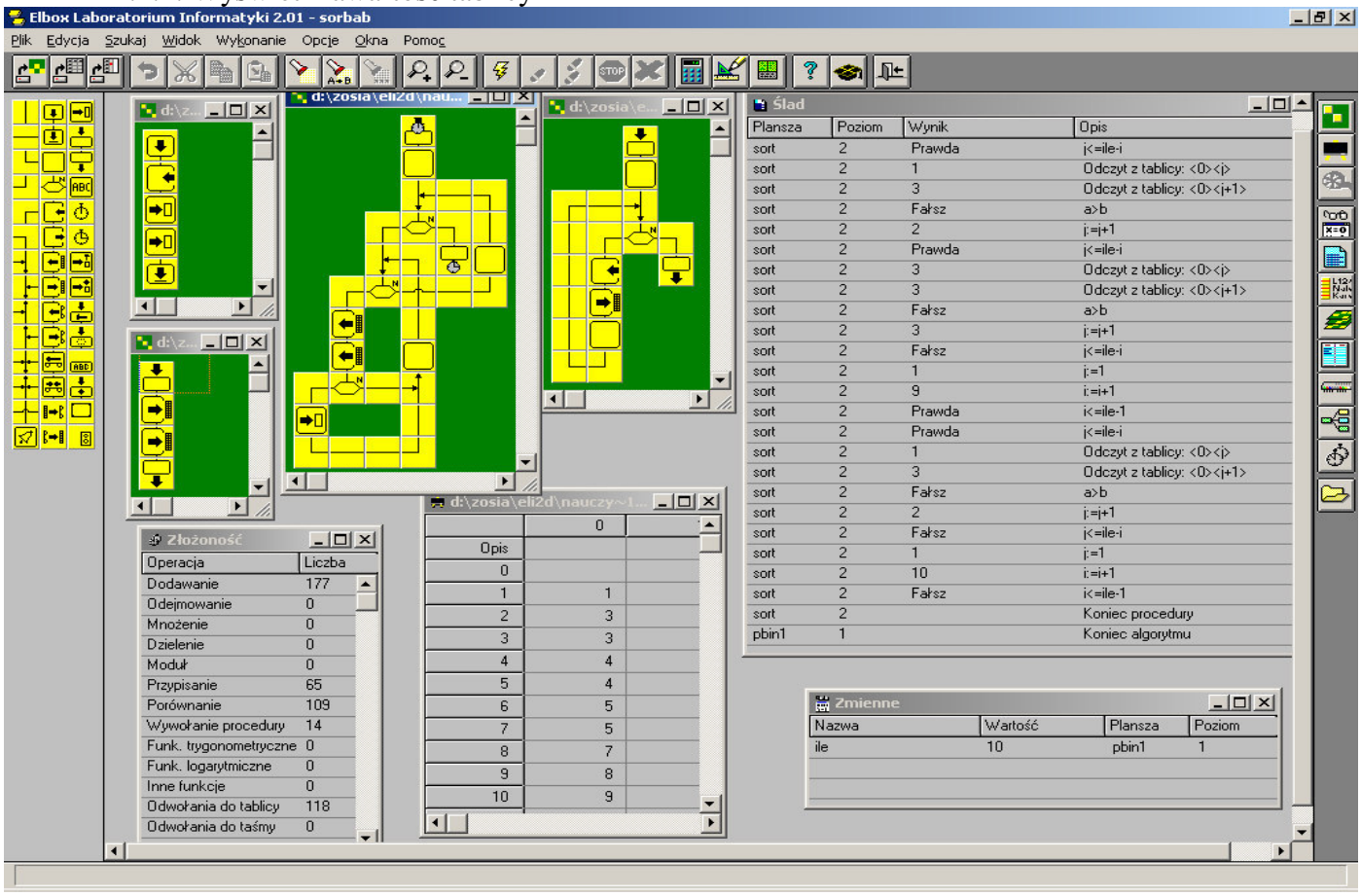
2.2.1. Podaj rozmiary tablicy

2.2.2. Wypełnij tablice danymi z klawiatury

2.2.3. Posortuj dane bąbelkowo – podano kod źródłowy w języku C++

```
void babelki(element tab[], long l, long p) // pusta lista parametrów- w metodzie atrybuty klasy są
  { for( long i =l; i<p; i++)                // dostępne bezpośrednio
        for (long j=l;j<p-i; j++)
            porownaj_zamien(tab[j], tab[j+1]);          // przekazanie indeksów tablicy j oraz j+1
   }
void  porownaj_zamien(element &a, element &b)  //  przekazanie indeksów tablicy
    { if (b<a)                  // porównanie bezpośrednie elementów tablicy, czyli tab[b]<tab[a] itd.
       zamien(a, b);
     }
void zamien(element &a, element &b)        //  przekazanie indeksów zamienianych elementów tablicy
     { element pom = a;                    //  zamiana elementów tablicy bezpośrednio
       a = b;                              //np. tab[a] = tab[b];
       b = pom;
      }
```

2.2.4. Wyświetl zawartość tablicy



2.3. Zdefiniuj klasę *Babelki*, która sortuje tablice elementów typu int. Użyj obiekt tej klasy ja w programie, którego algorytm działania podano w punkcie 2.2. Poszczególne operacje powinny być realizowane za pomocą nie statycznych metod obiektu klasy *Babelki*, ponieważ tablica jest atrybutem klasy. Podany kod w języku C++ należy przystosować do kodu w języku Java, traktując każdą z tych funkcji jako metody klasy *Babelki*. Oprócz tych metod należy uzupełnić klasę o metody, które wynikają z opisu z punktu 2.2 (*wypelnij_tablice, wyswietl_tablice*)