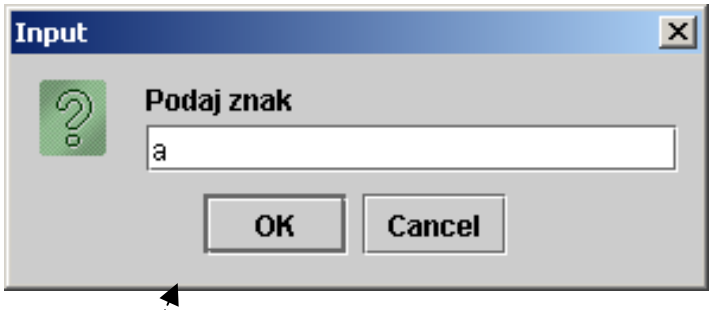
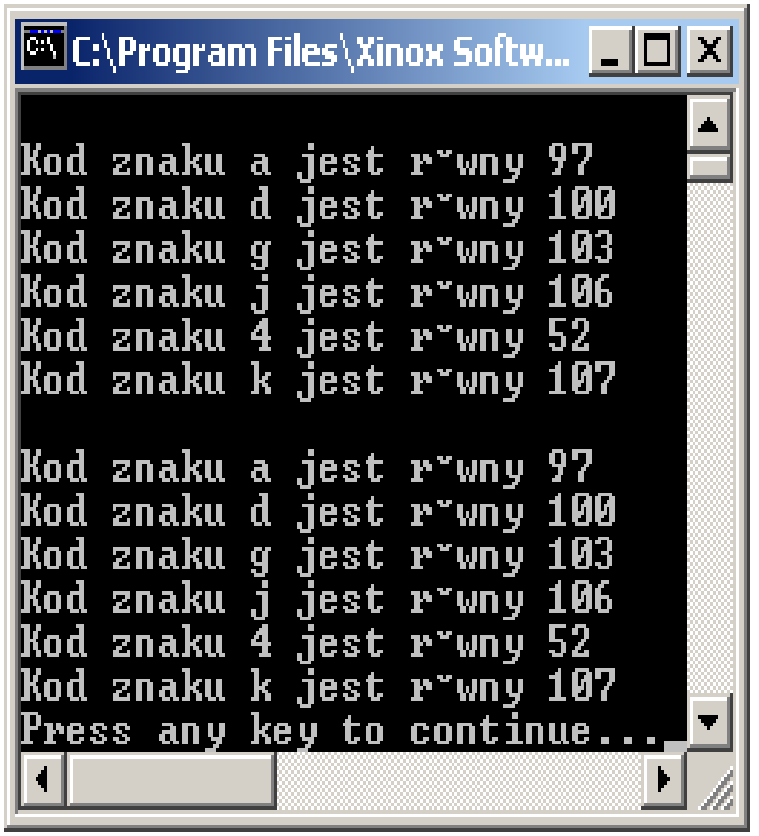
Przykład 1 – metody i atrybuty statyczne

```
import javax.swing.*;
import java.util.*;

public class Napis1
 { static String wynik;

    public static void Inicjuj
        {wynik = "";}

    public static void Dopisz_do_wyniku(char ch)
       { wynik+="\nKod znaku "+ ch + " jest równy "+ (int)ch;  }

    public static void Rysuj_graficznie()
       { JOptionPane.showMessageDialog(null, wynik);   }

    public static void Rysuj_konsolowo()
      { System.out.println(wynik);  }

    public static void main(String[] args)
     { String s;
       char ch ='a';
       Napis1.Inicjuj();
       while ( ch != 'k' )      //1 sposób podawania znaku
        { s=JOptionPane.showInputDialog(null, "Podaj znak");
          ch=s.charAt(0);
          Napis1.Dopisz_do_wyniku(ch);
         }
       Napis1.Rysuj_graficznie();
       Napis1.Rysuj_konsolowo();
       System.out.println(Napis1.wynik);
       System.exit(0);
    } }
import javax.swing.*;
```





```
import java.util.*;

public class Napis1
 {
   static String wynik;
    public static void Inicjuj()
        {wynik = "";}
    public static void Dopisz_do_wyniku(char ch)
       { wynik+="\nKod znaku "+ ch + " jest równy "+ (int)ch;  }
    public static void Rysuj_graficznie()
       { JOptionPane.showMessageDialog(null, wynik);   }
    public static void Rysuj_konsolowo()
      { System.out.println(wynik);  }
    public static void main(String[] args)
     { String s;
       char ch ='a';
       Napis1.Inicjuj();
       while ( ch != 'k' )
        { s=JOptionPane.showInputDialog(null, "Podaj znak");
          ch=s.charAt(0);
          Napis1.Dopisz_do_wyniku(ch);
        }
       Rysuj_graficznie();
       Napis1.Rysuj_konsolowo();
       System.out.println(Napis1.wynik);
       System.exit(0);
     }
 }
```

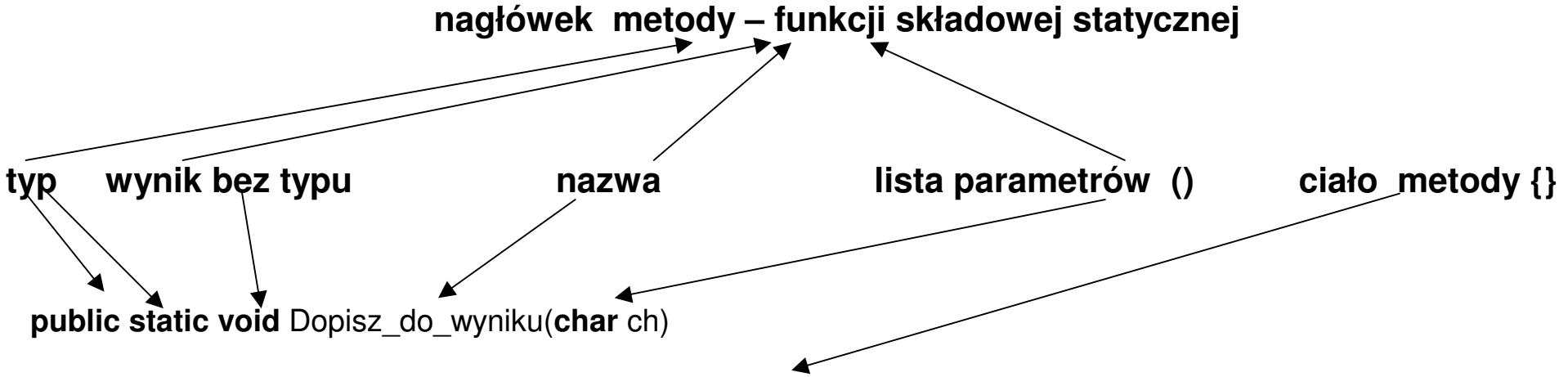
Nazwa klasy Napis1

Atrybut klasy typu **static** (składowa typu **static**) – istnieje przez cały czas działania programu w pamięci programu - **definicja atrybutu static**

Metody klasy Napis1 typu **static** (składowe funkcje klasy) – można je używać czyli wywoływać bez tworzenia obiektu klasy Napis1 – **definicje metod static**

Wywołania metod typu static klasy Napis1.

Wywołania atrybutu typu static klasy Napis1.



Uwaga: W metodzie typu static muszą być jedynie wywoływane atrybuty typu static i metody typu static

void – brak typu

Napis1.Rysuj_graficznie();

Wywołanie metody statycznej za pomocą nazwy klasy Napis1 w metodzie main. Jedynie w metodzie main dla metod typu **static** dodano do nazwy metody nazwę klasy Napis1 oraz operator wyboru „**.**”. **(Nie jest to obowiązkowe, czyli bez podania nazwy klasy metoda statyczna też może być wywołana w metodzie main własnej klasy)**

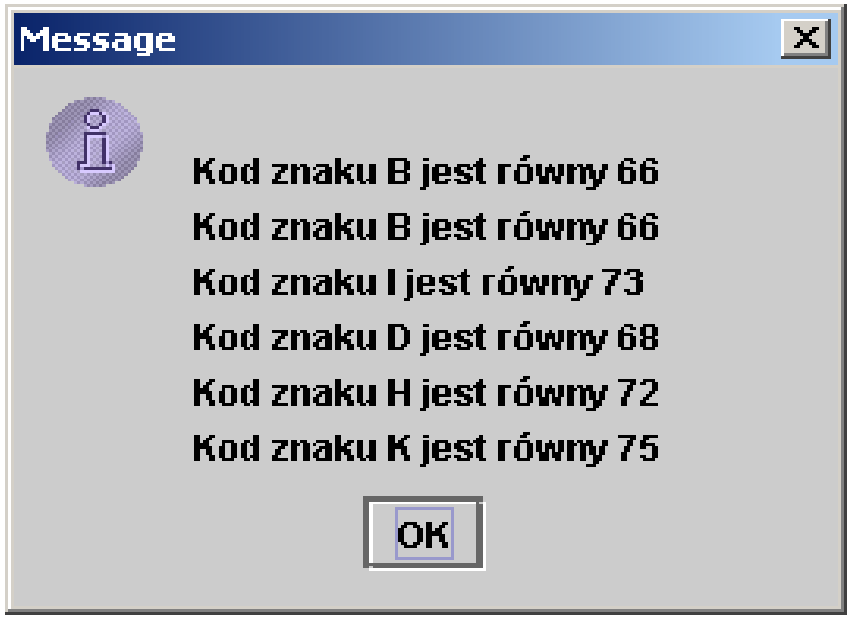
System.out.println(Napis1.wynik);

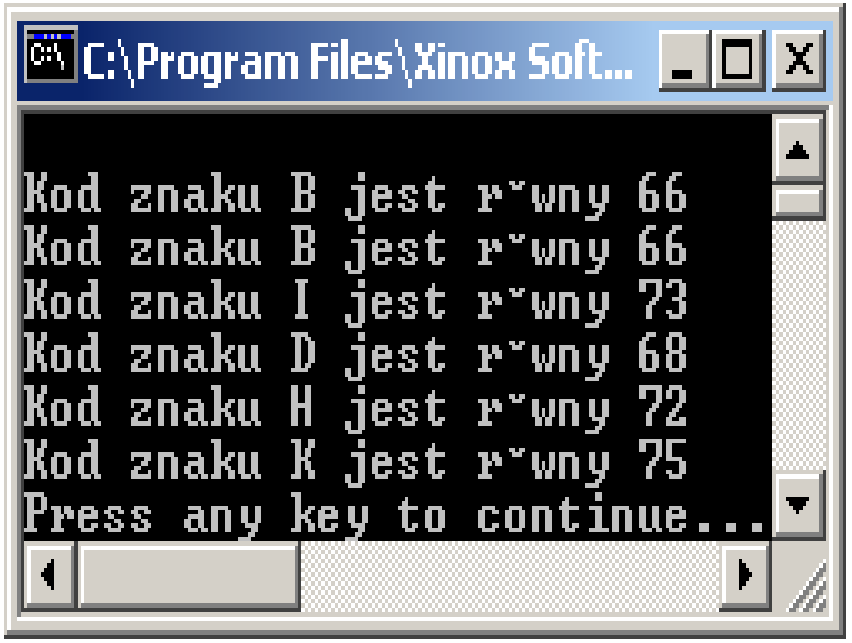
Wywołania atrybutu typu static klasy Napis1 w metodzie main. Jedynie w metodzie main dla atrybutów typu **static** dodano do nazwy metody nazwę klasy Napis1 oraz operator wyboru „**.**”**.(Nie jest to obowiązkowe, czyli bez podania nazwy klasy atrybut statyczny też może być wywołany w metodzie main własnej klasy)**

Obiektowy styl programowania - używanie metod w celu przetwarzania atrybutów obiektu - hermetyzacja

Przykład 2 – wielużywalność kodu klasy

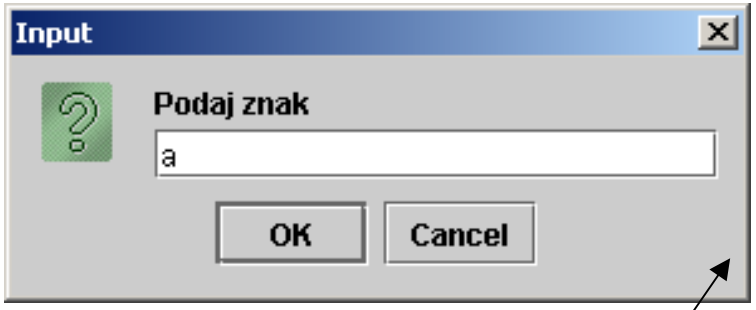
```
import javax.swing.*;
import java.util.*;
public class Napis2
 {
    static String wynik;
    public static void Inicjuj()
       {wynik = "";}
    public static void Dopisz_do_wyniku(char ch)
      {wynik+="\nKod znaku "+ ch + " jest równy "+ (int)ch;   }
   public static void Rysuj_graficznie()
      {JOptionPane.showMessageDialog(null, wynik);   }
   public static void Rysuj_konsolowo()
      { System.out.println(wynik);   }
  public static void main(String[] args)
    {
      char ch ='a';
      Napis2.Inicjuj();
      while ( ch != 'K' )
       {       //2 sposób podawania znaku
        ch= (char)(Math.random()*11 +65);  znaku
        Napis2.Dopisz_do_wyniku(ch);
       }
      Napis2.Rysuj_graficznie();
      Napis2.Rysuj_konsolowo();
      System.exit(0);
     }
 }
```

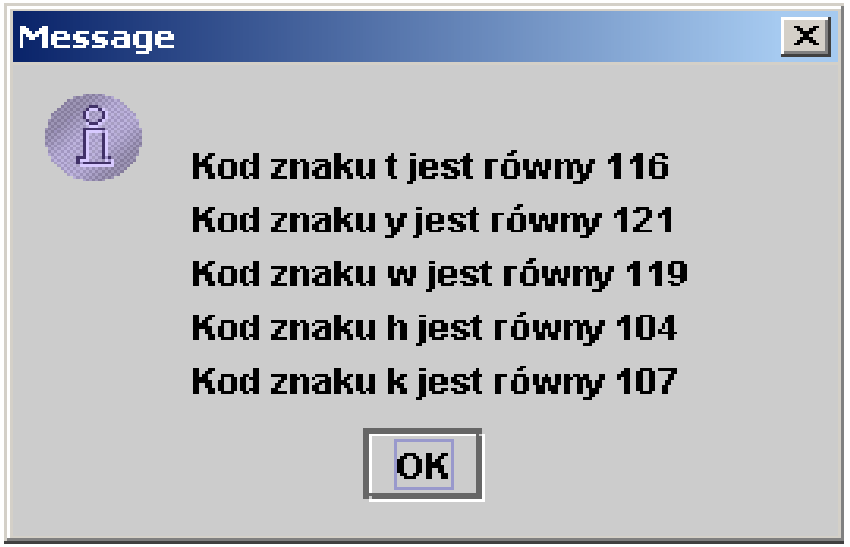


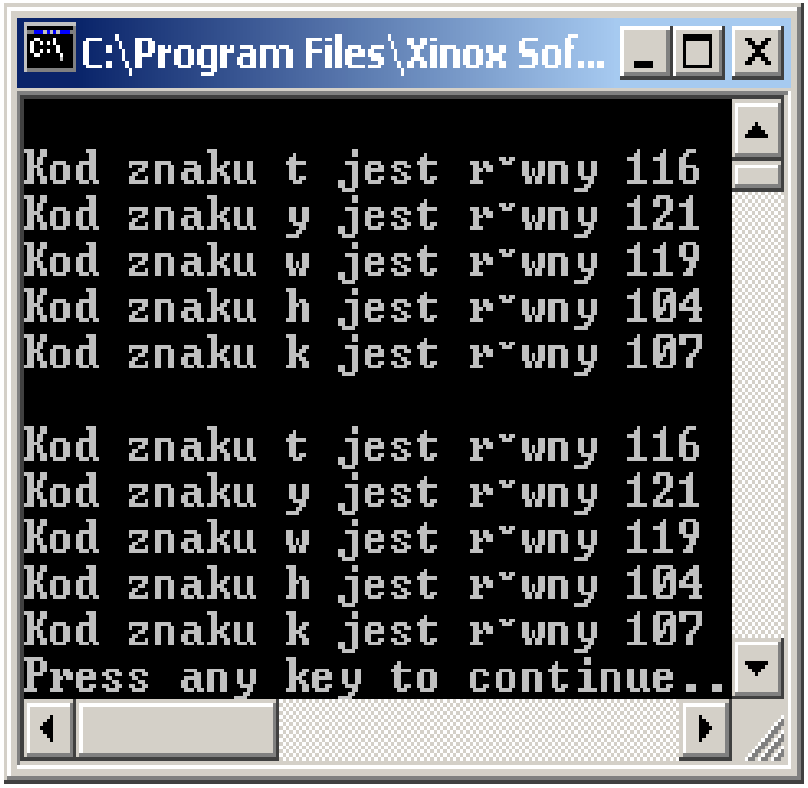


Przykład 3 – metody i atrybuty niestatyczne

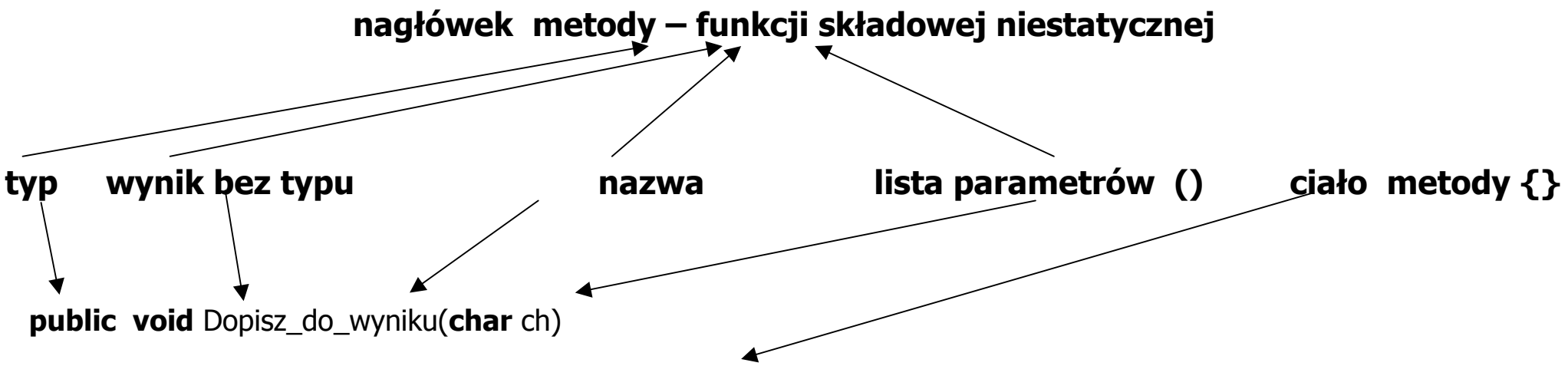
```
import javax.swing.*;
import java.util.*;

public class Napis3
 { String wynik;

   public void Inicjuj()
     { wynik = "";}

   public void Dopisz_do_wyniku(char ch)
     { wynik+="\nKod znaku "+ ch + " jest równy "+ (int)ch;}

   public void Rysuj_graficznie()
     {JOptionPane.showMessageDialog(null, wynik); }

   public void Rysuj_konsolowo()
     { System.out.println(wynik);}

   public static void main(String[] args)
    { String s;
      char ch ='a';
      Napis3 napis;
      napis= new Napis3();
      napis.Inicjuj();
      while ( ch != 'k' )
       { s =JOptionPane.showInputDialog(null, "Podaj znak");
          ch = s.charAt(0);
          napis.Dopisz_do_wyniku(ch);
        }
          napis.Rysuj_graficznie();
          napis.Rysuj_konsolowo();
          System.out.println(napis.wynik);
          System.exit(0);
    } }
```







Brak słowa **static** przy definicjach wymaga utworzenia obiektu typu Napis3.
Nazwa klasy Napis3 oznacza typ obiektu.
Obiekt powstaje po wywołaniu operatora **new**



Napis3 napis;

Referencja typu Napis3 niezainicjowana – brak obiektu typu Napis3

napis = **new** Napis3();

Referencja typu Napis3 zainicjowana – utworzono obiekt typu Napis3 za pomocą operatora **new**

napis.Rysuj_graficznie();

Wywołanie metody **niestatycznej** Rysuj_graficznie() za pomocą referencji **napis** do obiektu typu Napis3 oraz operatora wyboru „**.**”**-obowiązkowe w metodzie main własnej klasy dla metod niestatycznych**

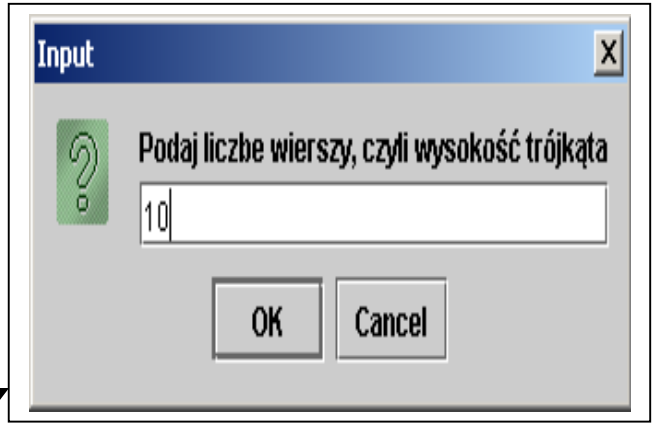
System.out.println(napis.wynik);

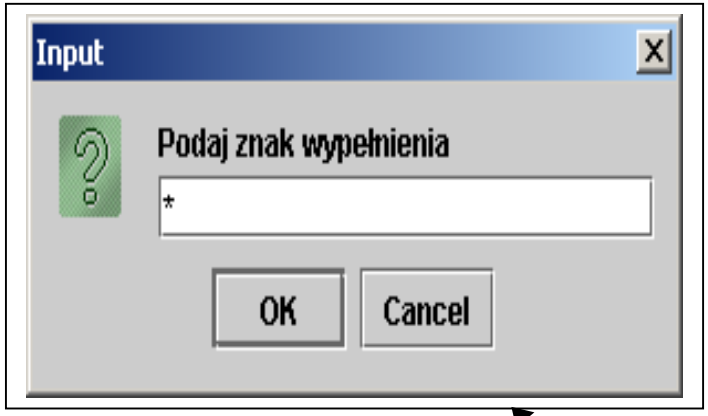
Wywołania atrybutu typu niestatycznego wynik klasy Napis1 w metodzie main za pomocą referencji **napis** do obiektu typu Napis3 oraz operatora wyboru „**.**” **– obowiązkowe w metodzie main własnej klasy dla atrybutów niestatycznych**

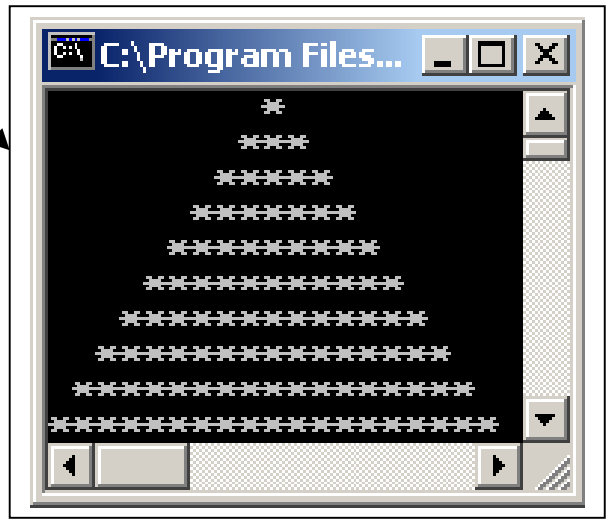
Przykład 4 - metody z instrukcją return

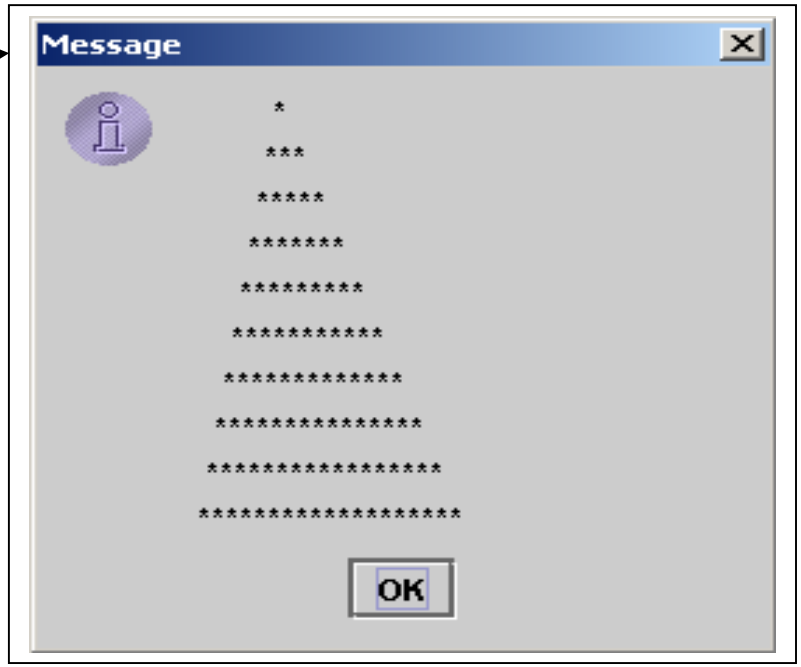
```
import javax.swing.*;
import java.util.*;
public class  Trojkat2
 { String rysunek;
   char wzor;
   int liczba_wierszy;
   void Inicjuj()
    { rysunek="";   }
   void Nadaj_wiersze(int wysokosc)
    { liczba_wierszy = wysokosc;}
   void Nadaj_wzor(char znak)
    { wzor =znak;  }
   int Liczba_spacji(int szerokosc)
    { return liczba_wierszy - szerokosc - 1;  }
   int Liczba_znakow (int szerokosc)
    { return 2*szerokosc + 1; }
  void Rysunek()
   { int i, j, liczba_spacji, liczba_znakow;
     for (j=0; j<liczba_wierszy; j++)
      {  liczba_spacji = Liczba_spacji(j);
         for (i=0; i<liczba_spacji; i++)
             rysunek += " ";
        liczba_znakow = Liczba_znakow(j);
        for (i = 0; i<liczba_znakow; i++)
           rysunek += wzor;
        rysunek += "\r\n";
      }
   }
```

```
void Narysuj_graficznie()
 { JOptionPane.showMessageDialog(null,rysunek); }

void Narysuj_konsolowo()
 { System.out.println(rysunek); }

public static void main(String[] args)
 { int w;
   String s;
   char z;
   Trojkat2 trojkat = new Trojkat2();
   trojkat.Inicjuj();
   s=JOptionPane.showInputDialog(null, "Podaj liczbe wierszy, czyli wysokość trójkąta");
   w =Integer.parseInt(s);
   trojkat.Nadaj_wiersze(w);
   s=JOptionPane.showInputDialog(null, "Podaj znak wypełnienia");
   z=s.charAt(0);
   trojkat.Nadaj_wzor(z);
   trojkat.Rysunek();
   trojkat.Narysuj_graficznie();
   trojkat.Narysuj_konsolowo();
   System.exit(0);
 }
}
```









nagłówek metody – funkcji składowej niestatycznej

wynik typu int **nazwa** **lista parametrów** **ciało metody**

```
int Liczba_spacji(int szerokosc)
 {
      return liczba_wierszy - szerokosc - 1;
 }
```

nagłówek metody – funkcji składowej niestatycznej

wynik bez typu **nazwa** **lista parametrów pusta ()** **ciało metody {}**

```
void Rysunek()
 { int i, j, liczba_spacji, liczba_znakow;
   for (j=0; j<liczba_wierszy; j++)
    {  liczba_spacji = Liczba_spacji(j);
       for (i=0; i<liczba_spacji; i++)
           rysunek += " ";
      liczba_znakow = Liczba_znakow(j);
      for (i = 0; i<liczba_znakow; i++)
         rysunek += wzor;
      rysunek += "\r\n";
    }
 }
```

Metoda klasy wywołana w innej metodzie tej klasy może być wywołana bez nazwy klasy. Wyjątkiem są wywołania metod niestatycznych danej klasy w jej metodzie main – przykład 3