Języki i metody programowania – Java
INF302W
Wykład 2 (część 2)

Autor

Dr inż. Zofia Kruczkiewicz

Struktura wykładu

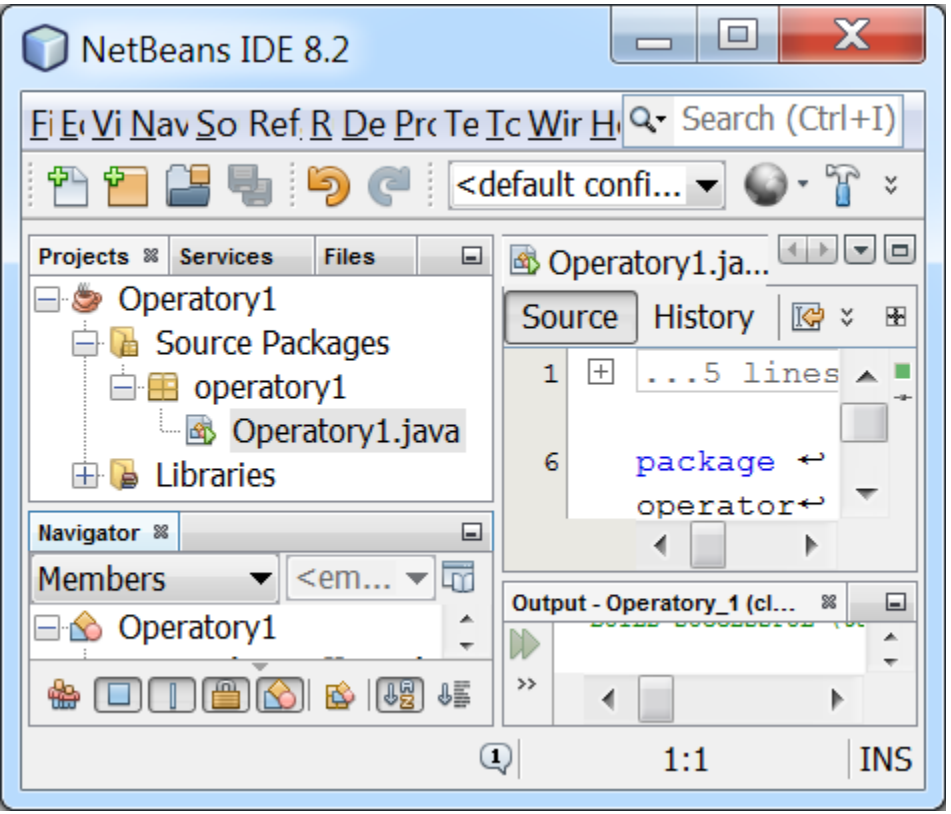
1. **Identyfikacja danych reprezentowanych przez klasy podczas opracowania koncepcji prostego programu obiektowego. Tworzenie programów z użyciem jednej i wielu klas: budowa klasy, konstruktory, metody, zastosowanie składowych statycznych i niestatycznych, operator new, odwołanie do obiektów-operator kropka, wywołanie metod, przeciążenie metod.**

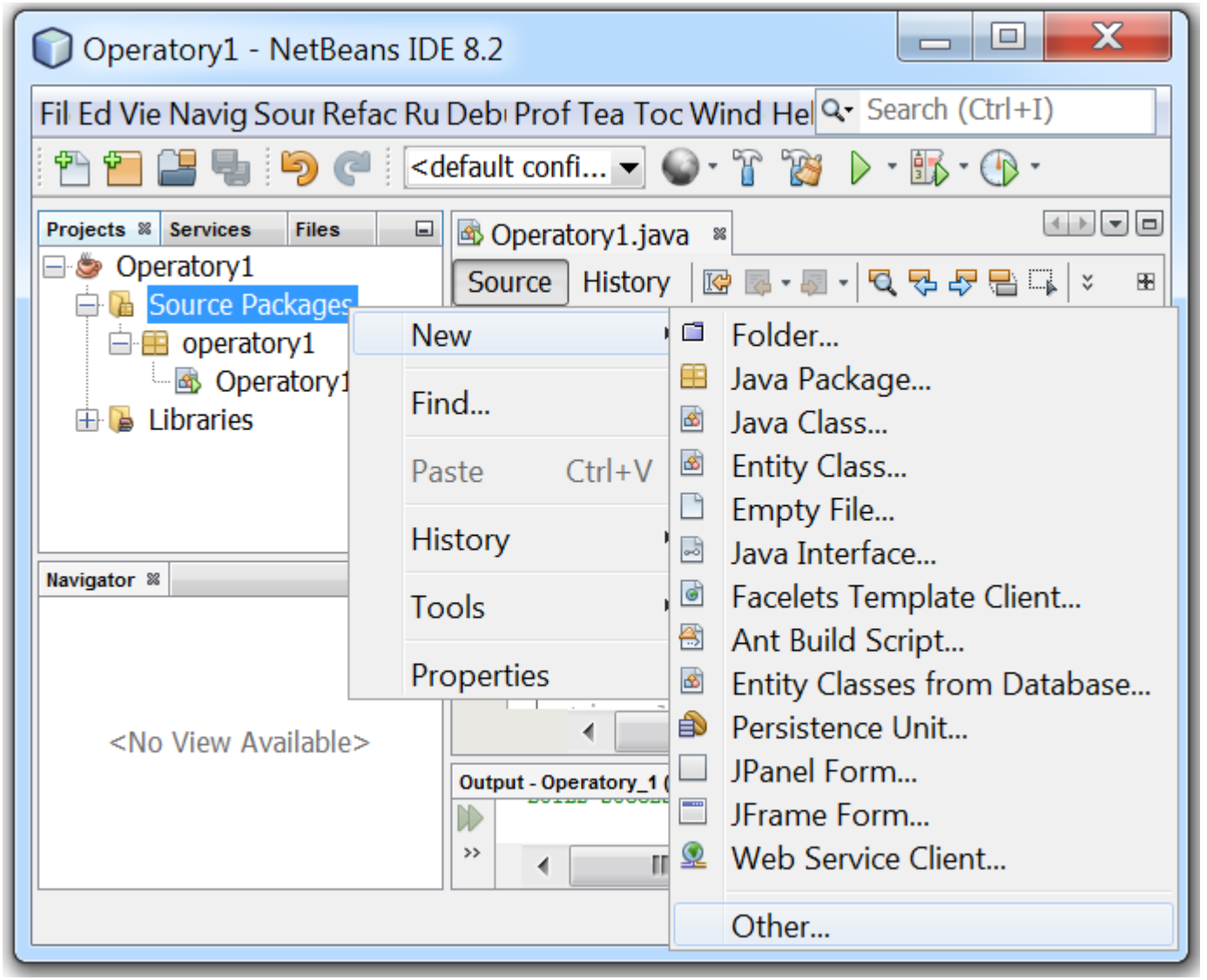
Operatory

1. Operatory jednoargumentowe + ,-. Operatory inkrementacji przedrostkowej i przyrostkowej – przykład identyfikacji klas (przykład z p.11 z wykładu java_wyklad1_2)

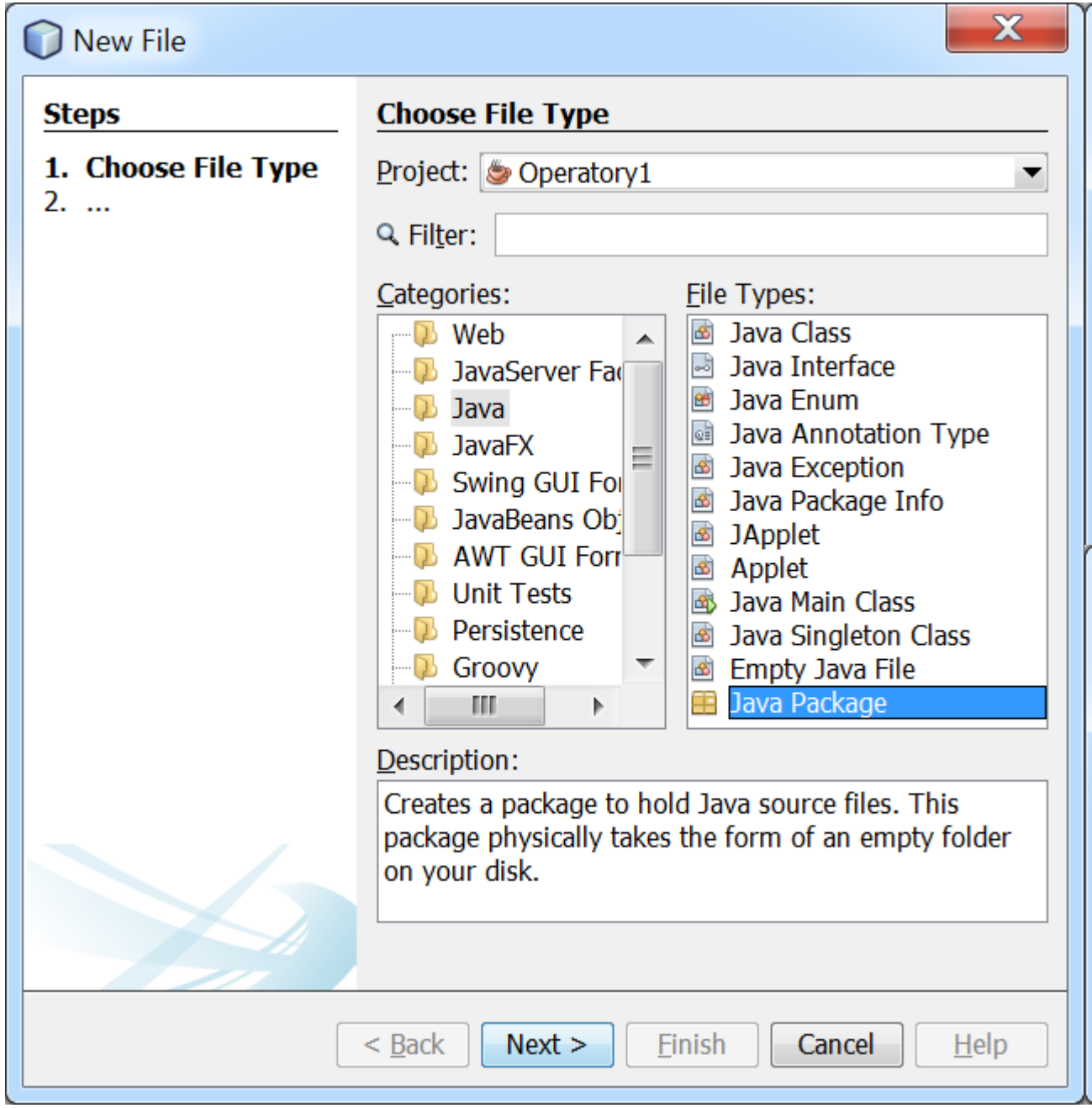
- Operatory jednoargumentowe – i plus służą do określenia wartości dodatniej lub ujemnej.
- Operatory inkrementacji i dekrementacji przedrostkowej np. ++i lub --i wykonują się najpierw, zanim wykona się wyrażenie, w którym użyto te operatory.
- Operatory inkrementacji i dekrementacji przyrostkowej np. i++ lub i-- wykonują się po wykonaniu wyrażenia, w którym użyto te operatory.

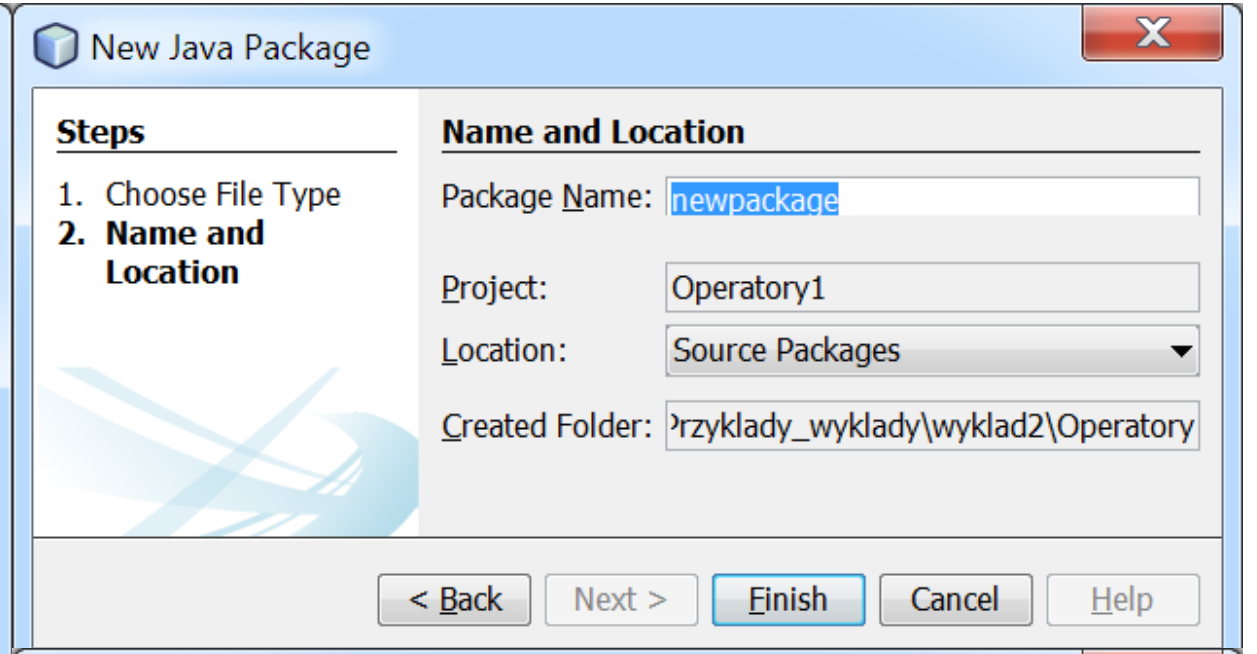
1.1. Przykład: Tworzenie pakietu operatory w nowym projekcie Operatory1 do przechowywania klas biznesowych. Projekt należy założyć wg pliku java_wyklad1_1.pdf

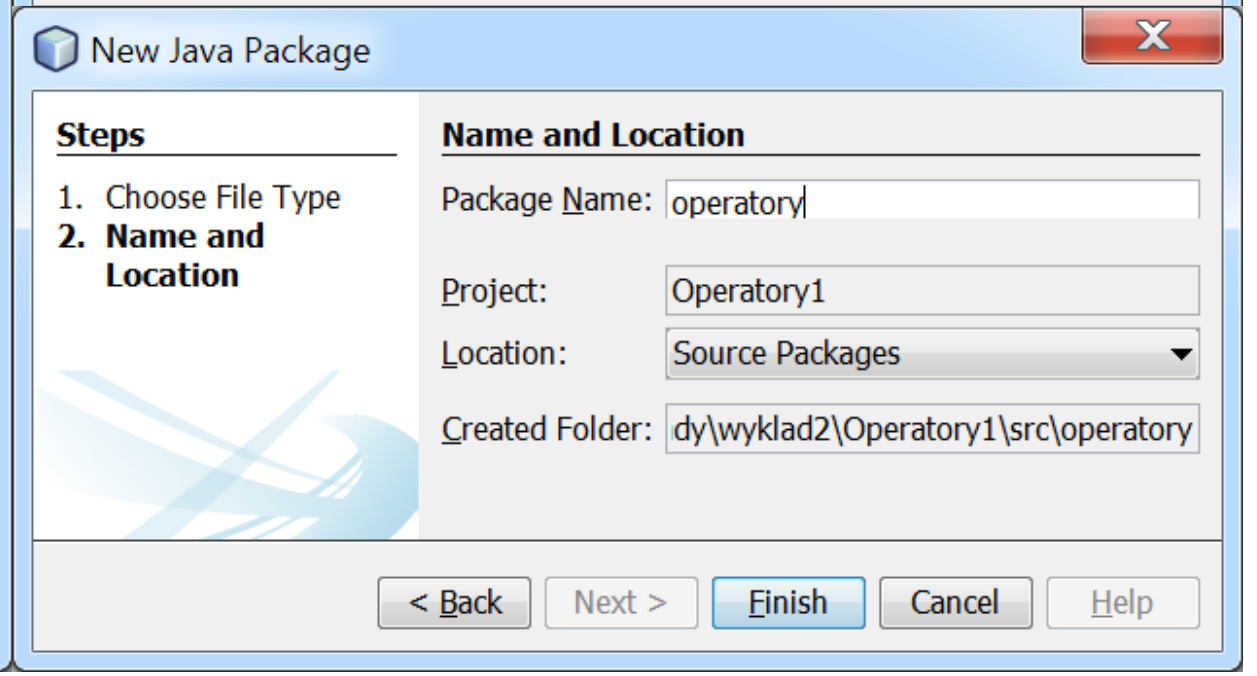




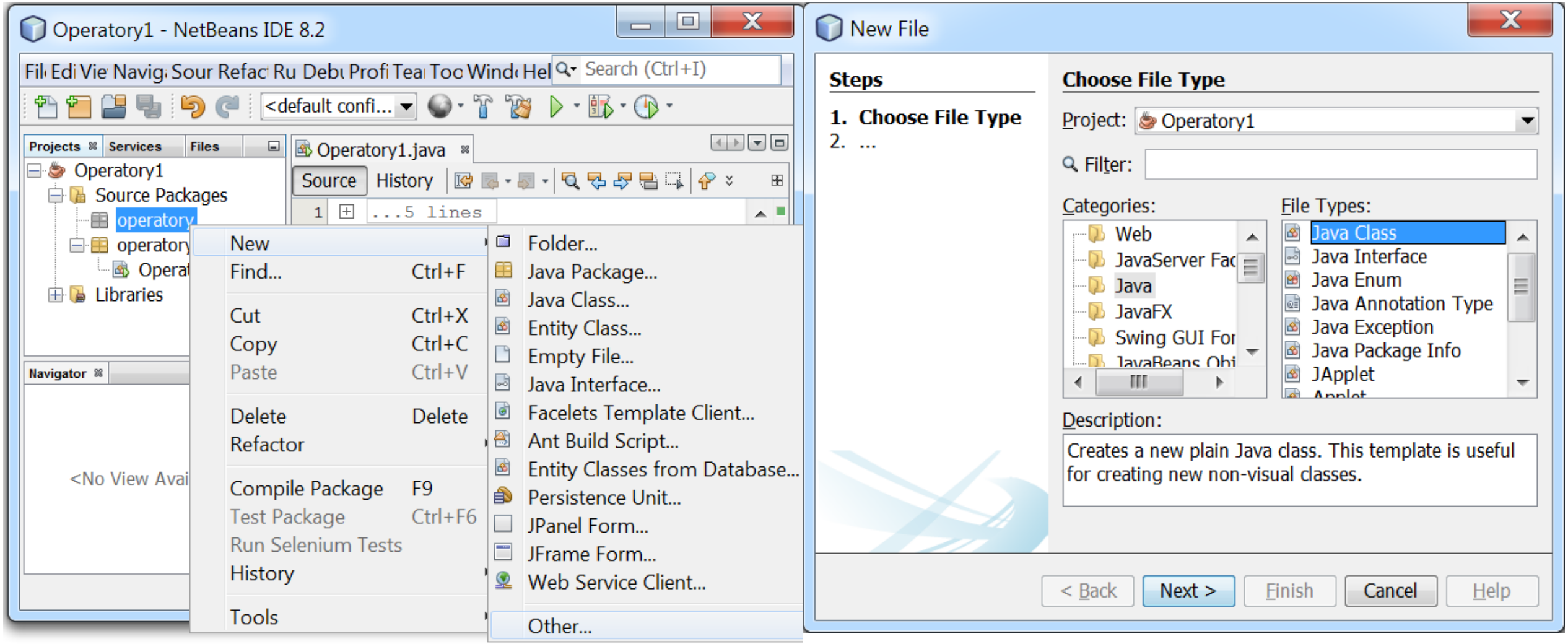
(cd) Tworzenie pakietu operatory do przechowywania klas biznesowych



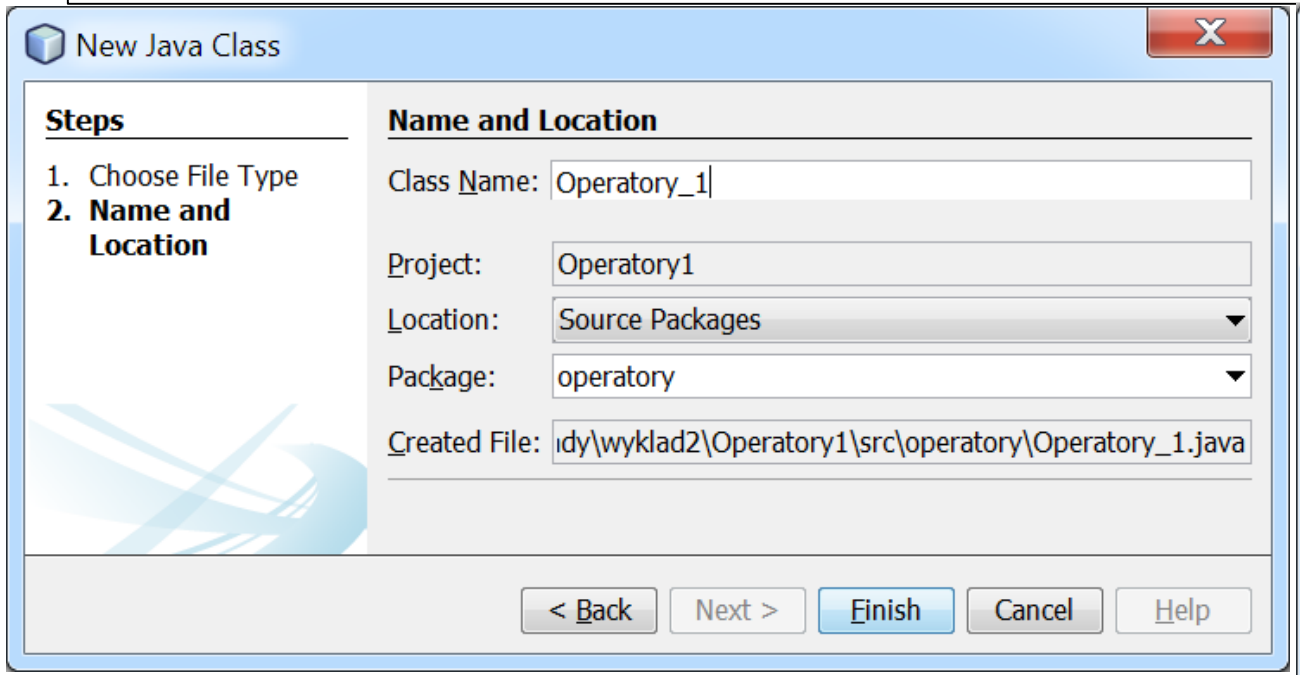


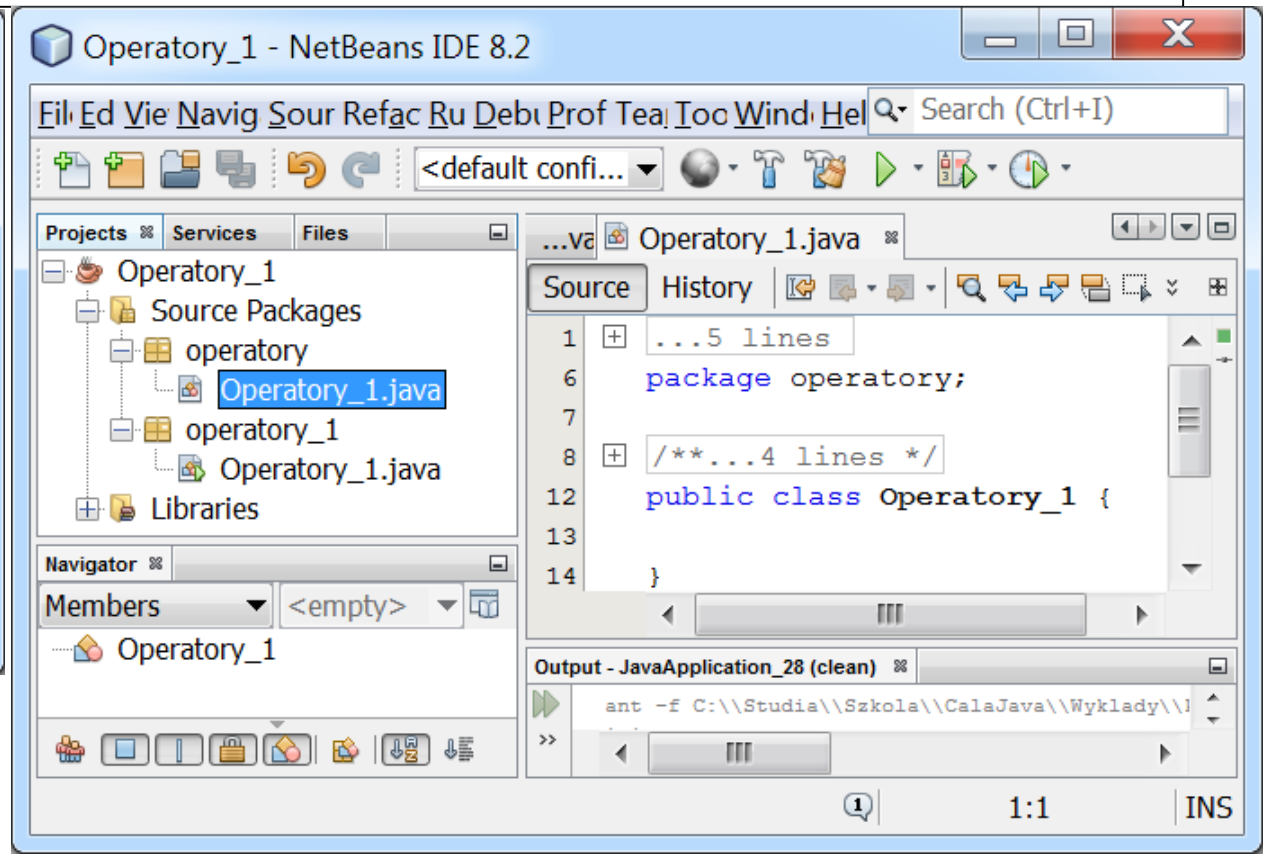


1.2. Tworzenie klasy Javy Operatory_1 w pakiecie operatory do przechowywania klas biznesowych



1.2. (cd) Tworzenie klasy Javy Operatory_1 w pakiecie operatory do przechowywania klas biznesowych - definicja klasy



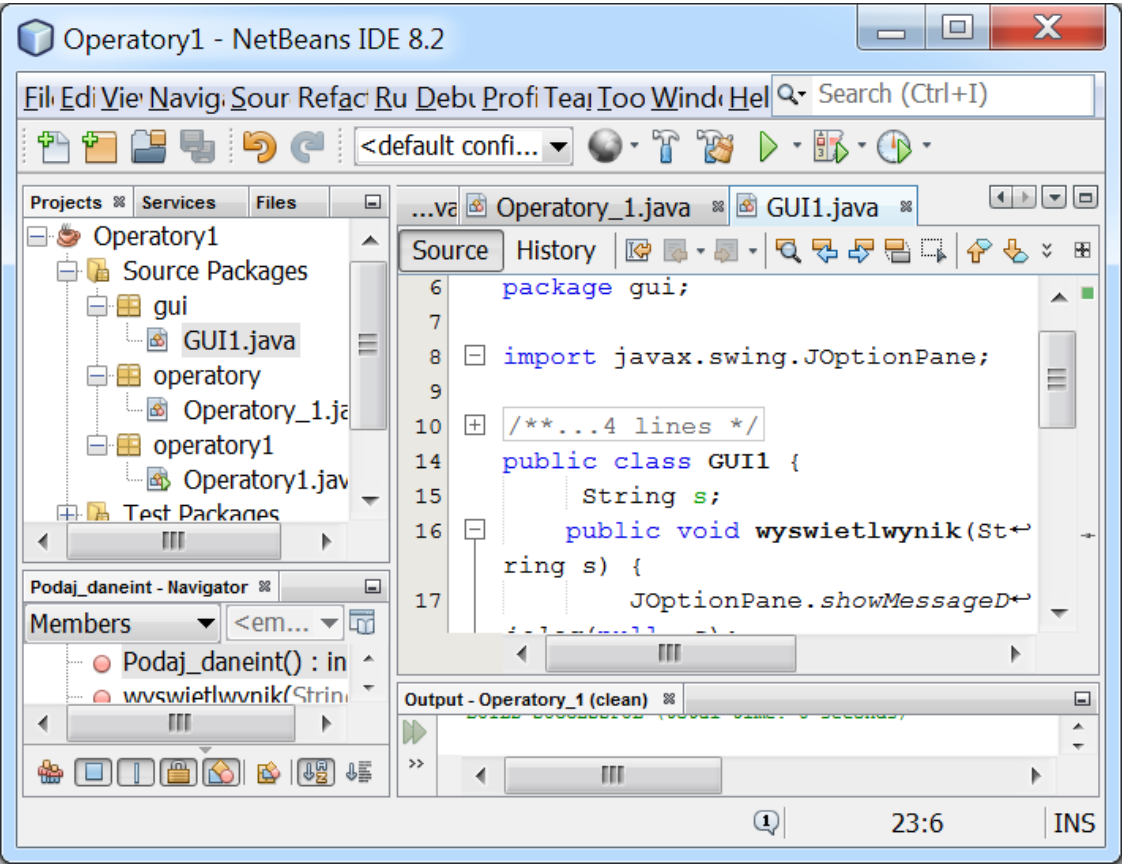


package operatory **//słowo kluczowe package**

```
public class Operatory_1 {          //klasa z logiką biznesową
    int arg11;                      //obsługująca operatory jednoargumentowe
    int wynik1;
    public void Przypisz_argument(int arg) {
        arg11 = arg;
    }
```

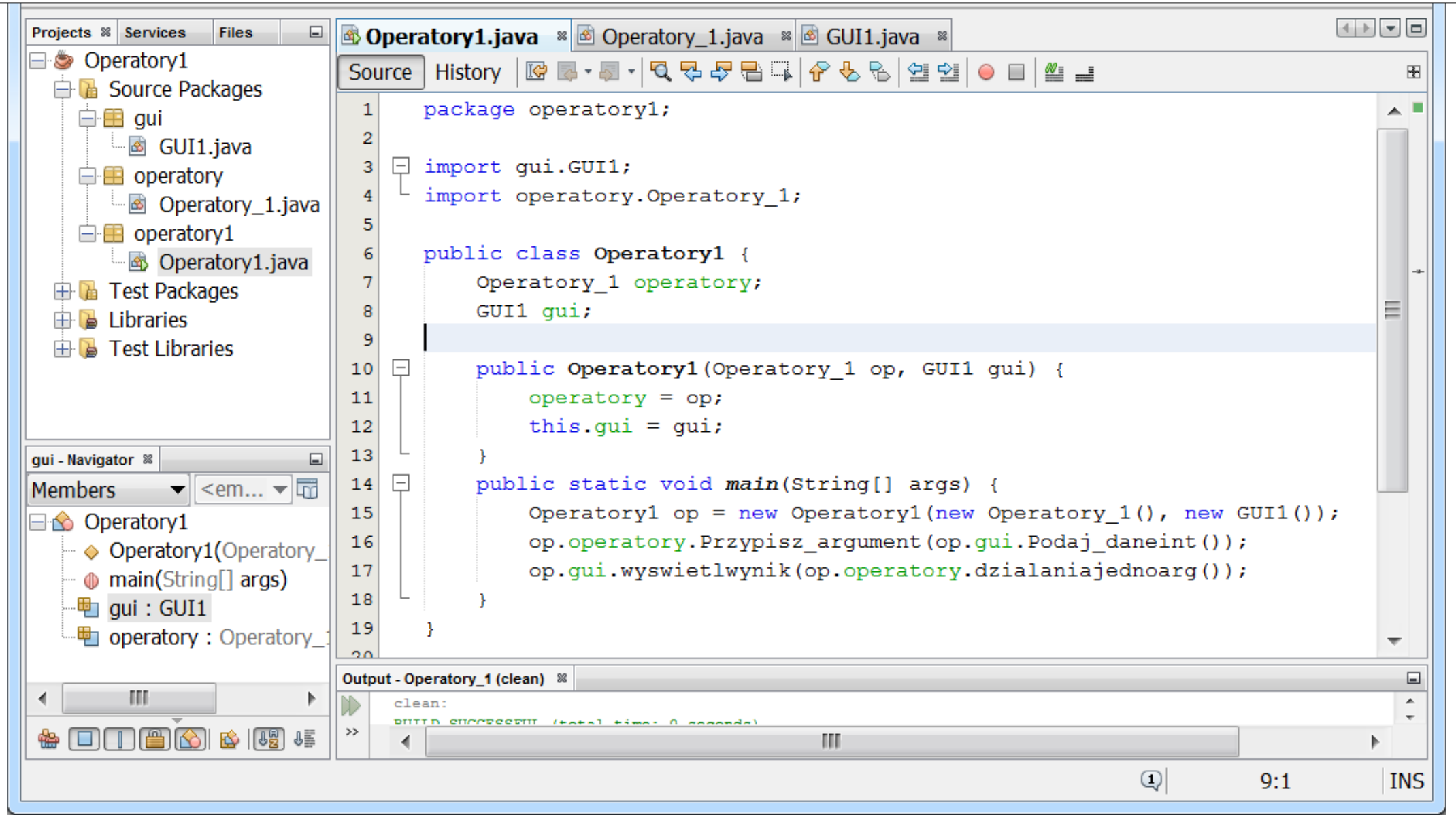
```
public String dzialaniajednoarg() {
        String s;
        s = "arg11 : " + arg11 + "\n";
        s += "++arg11 : " + ++arg11 + "\n";          // Pre-increment
                                                     // dodanie do łańcucha s nowego łańcucha s+= czyli s=s+
        s += "arg11++ : " + arg11++ + "\n";          // Post-increment
        s += "arg11 : " + arg11 + "\n";
        s += "—arg11 : " + --arg11 + "\n";           // Pre-decrement
        s += "arg11-- : " + arg11-- + "\n";          // Post-decrement
        s += "arg11 : " + arg11 + "\n";
        arg11 = -1;
        s += "\narg11 : " + arg11 + "\n";
        s += "++arg11 : " + ++arg11 + "\n";          // Pre-increment
        s += "arg11++ : " + arg11++ + "\n";          // Post-increment
        s += "arg11 : " + arg11 + "\n";
        s += "--arg11 : " + --arg11 + "\n";          // Pre-decrement
        s += "arg11-- : " + arg11-- + "\n";          // Post-decrement
        s += "arg11 : " + arg11 + "\n";
       return s;
    }
}
```

1.3. Utworzenie klasy Javy GUI1 w pakiecie gui do przechowywania klas interfejsu graficznego użytkownika (podobnie jak pakiet operatory i klasa Operatory_1)



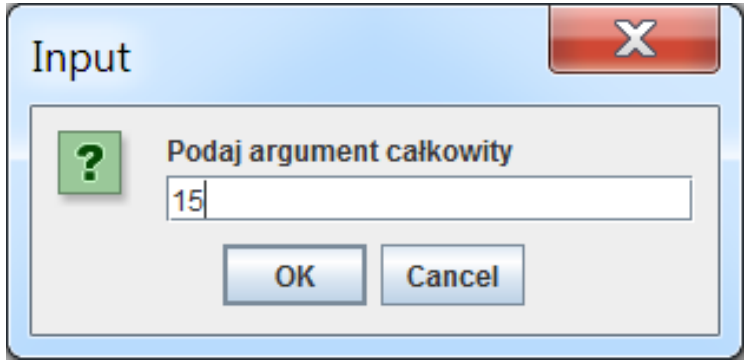

```
package gui;

import javax.swing.JOptionPane;

public class GUI1 {          //obsługa wprowadzania i
                             //prezentowania danych
    String s;
    public void wyswietlwynik(String s) {
        JOptionPane.showMessageDialog(null, s);
    }
    public int Podaj_daneint() {
     s =
  JOptionPane.showInputDialog(null, "Podaj argument całkowity");
        return Integer.parseInt(s);
    }
}
```

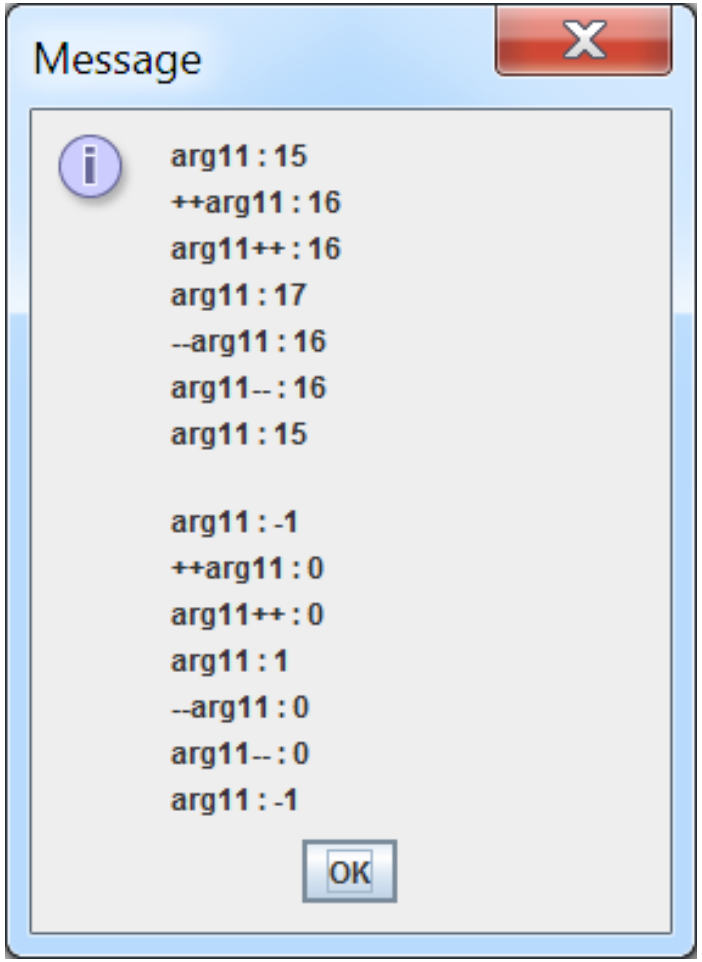
1.4. Zdefiniowanie klasy zarządzającej Operatory1 w pakiecie operatory1, utworzonych podczas tworzenia projektu (slajd 5) oraz uruchomienie programu



1.4(cd). Zdefiniowanie klasy zarządzającej Operatory1 w pakiecie operatory1, utworzonych podczas tworzenia projektu (slajd 5) oraz uruchomienie programu

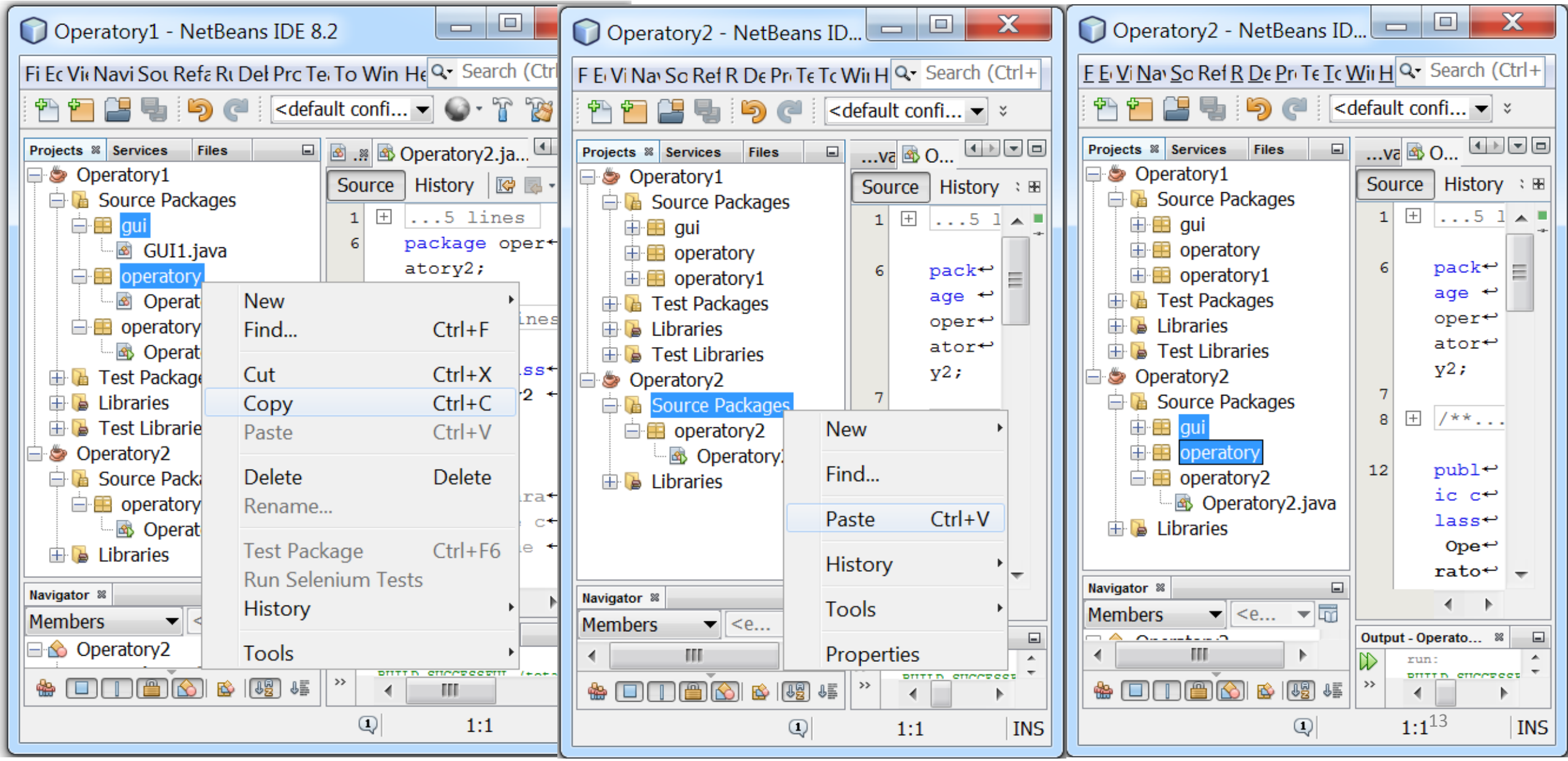
```
package operatory1;

import gui.GUI1;          //import kodu klasy GUI1 z pakietu gui
import operatory.Operatory_1; //import kodu klasy Operatory_1 z
                              //pakietu operatory

public class Operatory1 { // klasa zarządzająca
    Operatory_1 operatory;
    GUI1 gui;
    public Operatory1(Operatory_1 op, GUI1 gui) {
        operatory = op;
        this.gui = gui;
    }
    public static void main(String[] args) {
        Operatory1 op = new Operatory1(new Operatory_1(), new GUI1());
        op.operatory.Przypisz_argument(op.gui.Podaj_daneint());
        op.gui.wyswietlwynik(op.operatory.dzialaniajednoarg());
    }
}
```





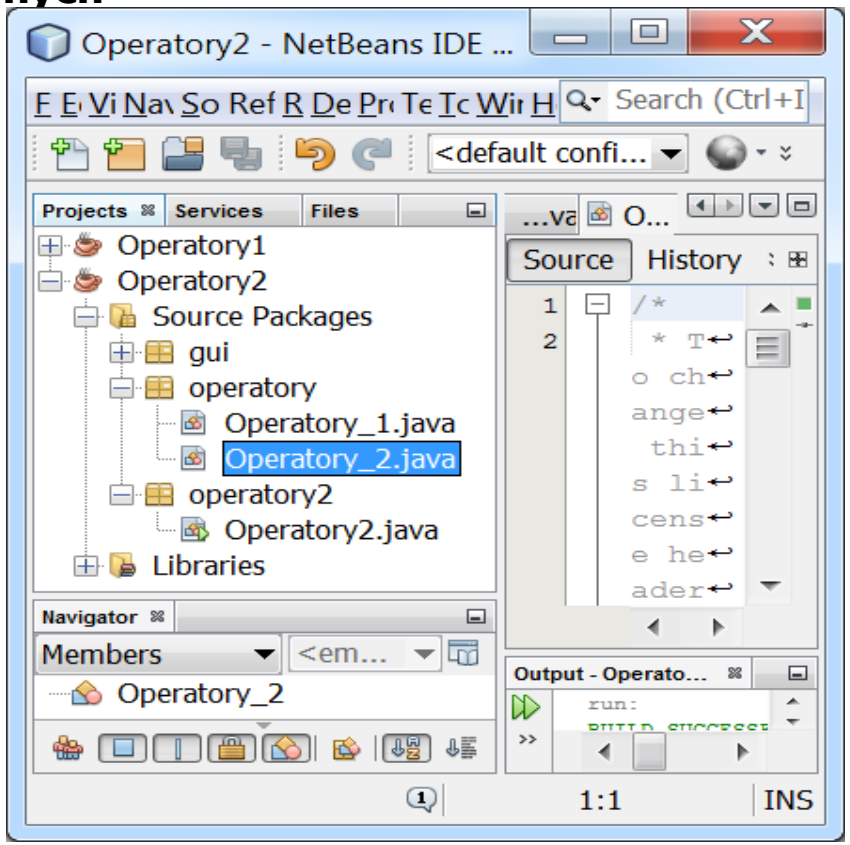
2. Operatory relacyjne

2.1. Utworzenie nowego projektu Operatory2 i skopiowanie do niego pakietów gui oraz operatory z projektu Operatory1



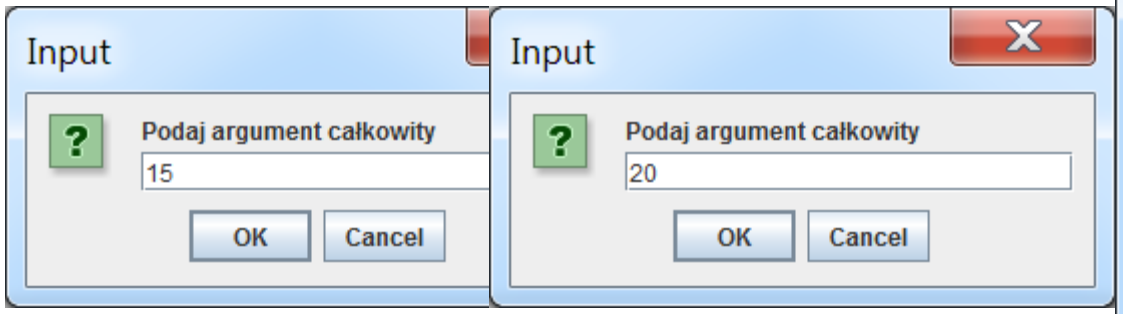
2.2. Definicja nowej klasy Operatory_2 w pakiecie operatory

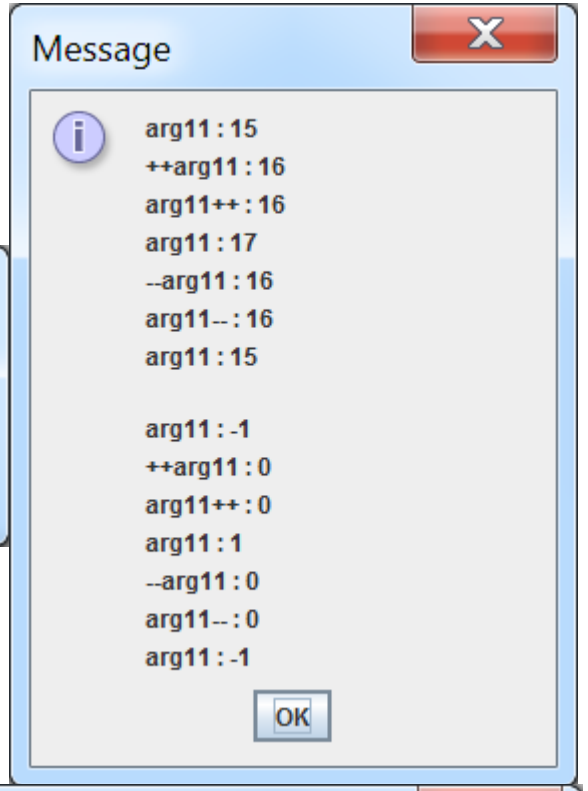
```
package operatory;

public class Operatory_2  extends Operatory_1{   //klasa z logiką biznesową z dodaną
                                                  //obsługą operatów relacyjnych
   int arg12;

  public void Przypisz_argumentyint(int arg1, int arg2) {
      this.arg11 = arg1;
      this.arg12 = arg2;
   }
  public String dzialaniarelacyjne() {
      String s;
      s = "arg11 = " + arg11 + "\n";
      s += "arg12 = " + arg12 + "\n";
      s += "arg11 > arg12 jest " + (arg11 > arg12) + "\n";
      s += "arg11 < arg12 jest " + (arg11 < arg12) + "\n";
      s += "arg11 >= arg12 jest " + (arg11 >= arg12) + "\n";
      s += "arg11 <= arg12 jest " + (arg11 <= arg12) + "\n";
      s += "arg11 == arg12 jest " + (arg11 == arg12) + "\n";
      s += "arg11 != arg12 jest " + (arg11 != arg12) + "\n";
      s += "(arg11 < 10) && (arg12 < 10) jest " + ((arg11 < 10) && (arg12 < 10)) + "\n";
      s += "(arg11 < 10) || (arg12 < 10) jest " + ((arg11 < 10) || (arg12 < 10)) + "\n";
      return s;  }
  }
```



2.3. Zdefiniowanie klasy zarządzającej Operatory2 w pakiecie operatory2, utworzonych podczas tworzenia projektu oraz uruchomienie programu

```
package operatory2;

import gui.GUI1;
import operatory.Operatory_2;

public class Operatory2 {
  Operatory_2 operatory;
  GUI1 gui; //obsługa wprowadzania i prezentowania danych – klasa bez zmian

  public Operatory2(Operatory_2 op, GUI1 gui) {
        operatory = op;
        this.gui = gui;
    }

 public static void main(String[] args) {
   Operatory2 op = new Operatory2(new Operatory_2(), new GUI1());
   op.operatory.Przypisz_argumentyint(op.gui.Podaj_daneint(),
                                      op.gui.Podaj_daneint());
   op.gui.wyswietlwynik(op.operatory.dzialaniajednoarg());
   op.gui.wyswietlwynik(op.operatory.dzialaniarelacyjne()); }
}
```



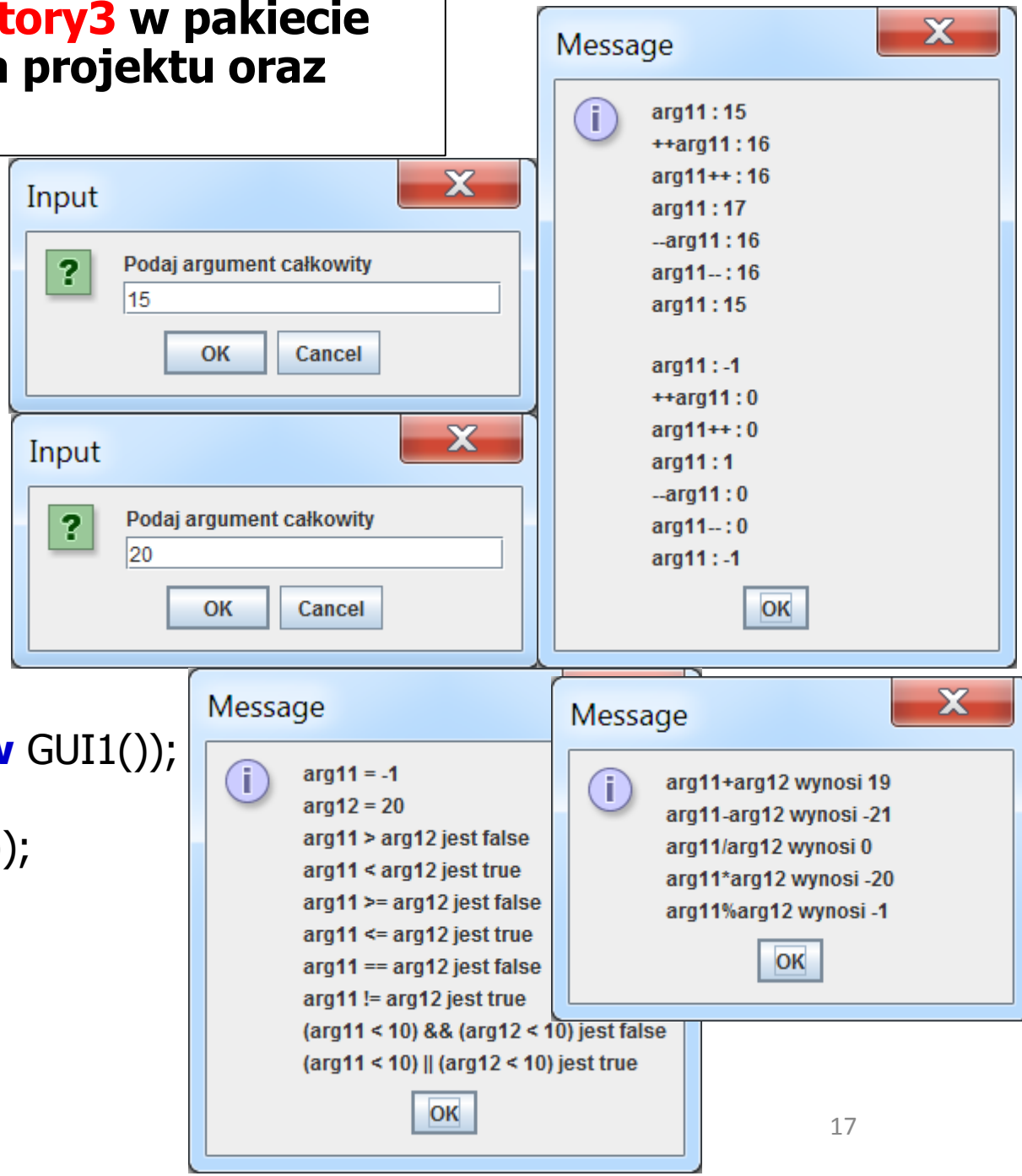


3. Operatory arytmetyczne dla argumentów typu int
3.1. Utworzenie nowego projektu Operatory3 i skopiowanie do niego pakietów gui oraz operatory z projektu Operatory2.
3.2. Wykonanie klasy Operatory_3 w pakiecie operatory i definicja tej klasy

```
package operatory;

public class Operatory_3  extends Operatory_2 { //klasa z logiką biznesową z dodaną obsługą
                                                 //operatorów arytmetycznych dla argumentów typu int
   public String dzialaniaint() {
       String s = "";
       wynik1 = arg11 + arg12;
       s = "arg11+arg12 wynosi " + wynik1 + "\n";
       wynik1 = arg11 - arg12;
       s += "arg11-arg12 wynosi " + wynik1 + "\n";
       wynik1 = arg11 / arg12;
       s += "arg11/arg12 wynosi " + wynik1 + "\n";
       wynik1 = arg11 * arg12;
       s += "arg11*arg12 wynosi " + wynik1 + "\n";
       wynik1 = arg11 % arg12;
       s += "arg11%arg12 wynosi " + wynik1 + "\n";
       return s;
    }
}
```

3.3. Zdefiniowanie klasy zarządzającej Operatory3 w pakiecie operatory3, utworzonych podczas tworzenia projektu oraz uruchomienie programu

```
package operatory3;

public class Operatory3 {//nowa klasa zarządzająca


    Operatory_3 operatory;
    GUI1 gui;

    public Operatory3(Operatory_3 op, GUI1 gui) {
        operatory = op;
        this.gui = gui;
    }

public static void main(String[] args) {
 Operatory3 op = new Operatory3(new Operatory_3(), new GUI1());
 op.operatory.Przypisz_argumentyint(op.gui.Podaj_daneint(),
                                    op.gui.Podaj_daneint());
   op.gui.wyswietlwynik(op.operatory.dzialaniajednoarg());
   op.gui.wyswietlwynik(op.operatory.dzialaniarelacyjne());
   op.gui.wyswietlwynik(op.operatory.dzialaniaint());
  }
}
```



4. Operatory arytmetyczne oraz przypisania dla argumentów typu float
4.1. Utworzenie nowego projektu Operatory4 i skopiowanie do niego pakietów gui oraz operatory z projektu Operatory3.
4.2. Wykonanie klasy Operatory_4 w pakiecie operatory i definicja tej klasy

```
package operatory;
class Operatory_4 extends Operatory_3 {          //klasa z logiką biznesową z dodaną obsługą operatorów
                                                 //arytmetycznych dla argumentów typu float
   float arg21, arg22;
   float wynik2;

   public void Przypisz_argumentyfloat(float arg1, float arg2) {
      this.arg21 = arg1;
      this.arg22 = arg2;
   }
   public String dzialaniafloat() {
       String s = "";
       wynik2 = arg21 + arg22;
       s = "arg21+arg22 wynosi " + wynik2 + "\n";
       wynik2 = arg21 - arg22;
       s += "arg21-arg22 wynosi " + wynik2 + "\n";
       wynik2 = arg21 / arg22;
       s += "arg21/arg22 wynosi " + wynik2 + "\n";
       wynik2 = arg21 * arg22;
       s += "arg21*arg22 wynosi " + wynik2 + "\n";
      return s;
   }
```

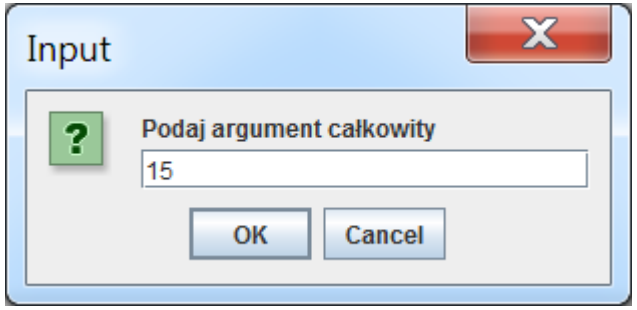
```
public String dzialaniaprzypisania() {
        String s = "";
        wynik2 += arg21;
        s += "wynik2 += arg21  wynosi " + wynik2 + "\n";
        wynik2 -= arg21;
        s += "wynik2 -= arg21  wynosi " + wynik2 + "\n";
        wynik2 *= arg21;
        s += "wynik2 *= arg21  wynosi " + wynik2 + "\n";
        wynik2 /= arg21;
        s += "wynik2 /= arg21  wynosi " + wynik2 + "\n";
        return s;
    }
}
```

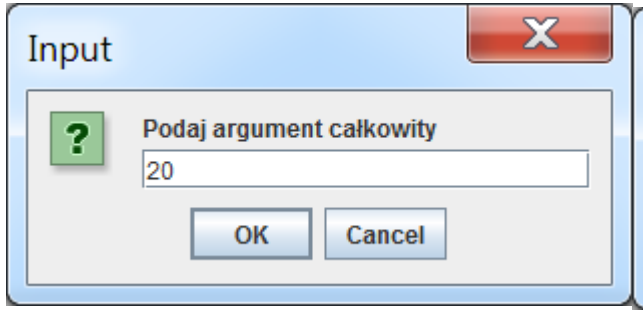
4.3. Utworzenie klasy Javy GUI1 w pakiecie gui do przechowywania klas interfejsu graficznego użytkownika (podobnie jak pakiet operatory i klasa Operatory_1)

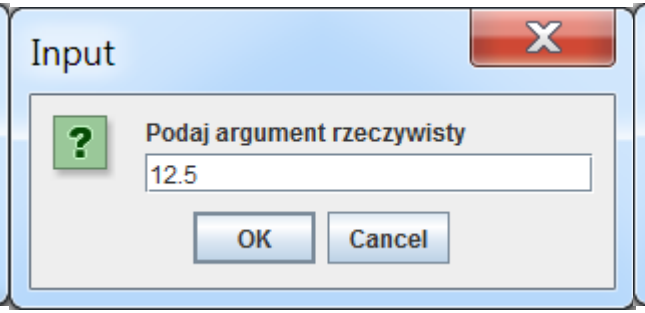
```
package gui;

import javax.swing.JOptionPane;

public class GUI2 extends GUI1 {          //klasa GUI rozszerzająca funkcjonalność
                                          //wprowadzania danych

    public float Podaj_danefloat() {
        s = JOptionPane.showInputDialog(null,"Podaj argument rzeczywisty");
        return Float.parseFloat(s);
    }
}
```

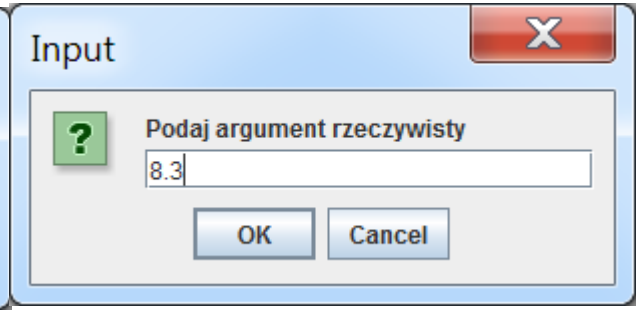
4.3. Zdefiniowanie klasy zarządzającej Operatory4 w pakiecie operatory4, utworzonych podczas tworzenia projektu oraz uruchomienie programu

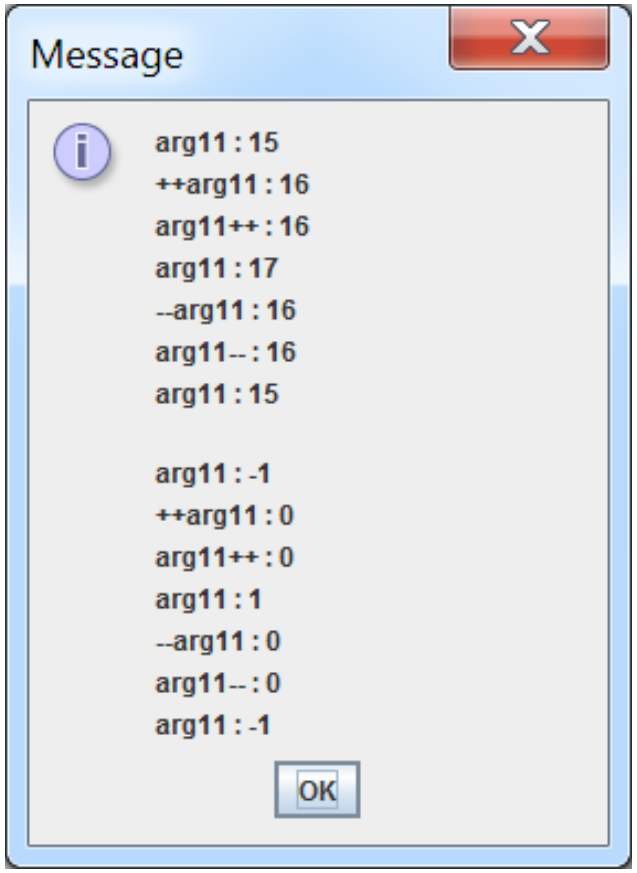
```
package operatory4;

import gui.GUI2;
import operatory.Operatory_4;

public class Operatory4 {   //nowa klasa zarządzająca
    Operatory_4 operatory;
    GUI2 gui;
    public Operatory4(Operatory_4 op, GUI2 gui) {
        operatory = op;
        this.gui = gui;
    }
    public static void main(String[] args) {
        Operatory4 op = new Operatory4(new Operatory_4(), new GUI2());
        op.operatory.Przypisz_argumentyint(op.gui.Podaj_daneint(), op.gui.Podaj_daneint());
        op.operatory.Przypisz_argumentyfloat(op.gui.Podaj_danefloat(), op.gui.Podaj_danefloat());
        op.gui.wyswietlwynik(op.operatory.dzialaniajednoarg());
        op.gui.wyswietlwynik(op.operatory.dzialaniarelacyjne());
        op.gui.wyswietlwynik(op.operatory.dzialaniaint());
        op.gui.wyswietlwynik(op.operatory.dzialaniafloat());
        op.gui.wyswietlwynik(op.operatory.dzialaniaprzypisania());  }
}
```


Input
Podaj argument całkowity
15
OK
Cancel

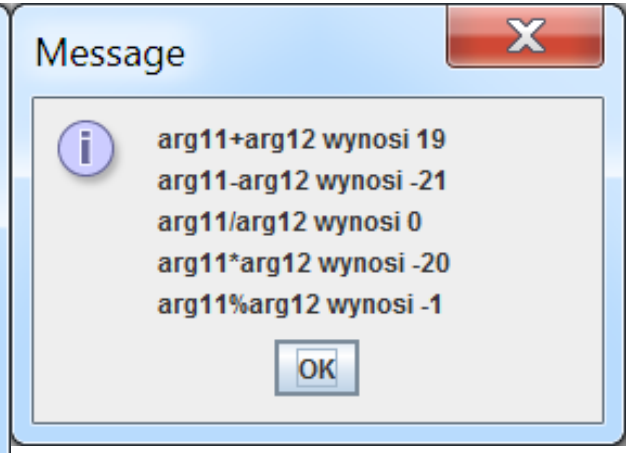

Input
Podaj argument całkowity
20
OK
Cancel

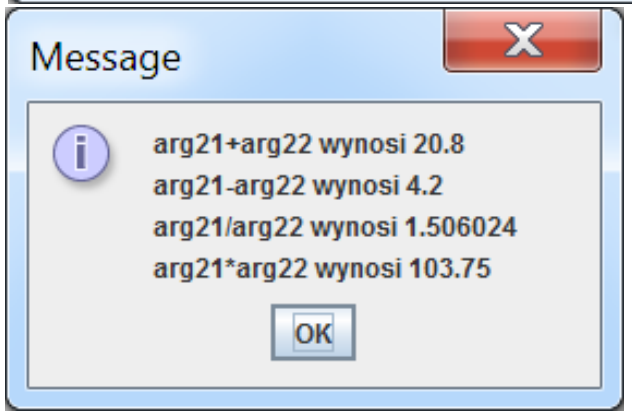

Input
Podaj argument rzeczywisty
12.5
OK
Cancel

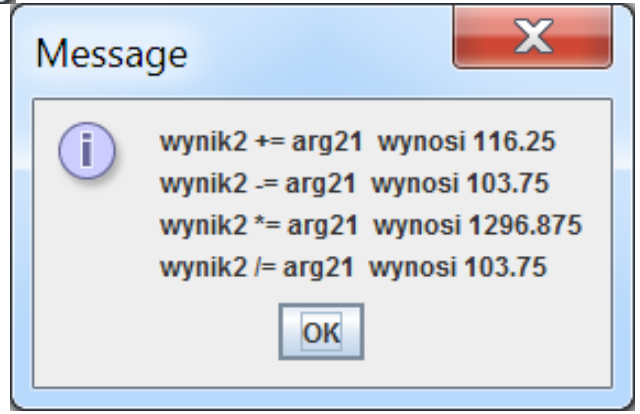

Input
Podaj argument rzeczywisty
8.3
OK
Cancel


Message
arg11 : 15
++arg11 : 16
arg11++ : 16
arg11 : 17
--arg11 : 16
arg11-- : 16
arg11 : 15
arg11 : -1
++arg11 : 0
arg11++ : 0
arg11 : 1
--arg11 : 0
arg11-- : 0
arg11 : -1
OK


Message
arg11 = -1
arg12 = 20
arg11 > arg12 jest false
arg11 < arg12 jest true
arg11 >= arg12 jest false
arg11 <= arg12 jest true
arg11 == arg12 jest false
arg11 != arg12 jest true
(arg11 < 10) && (arg12 < 10) jest false
(arg11 < 10) || (arg12 < 10) jest true
OK

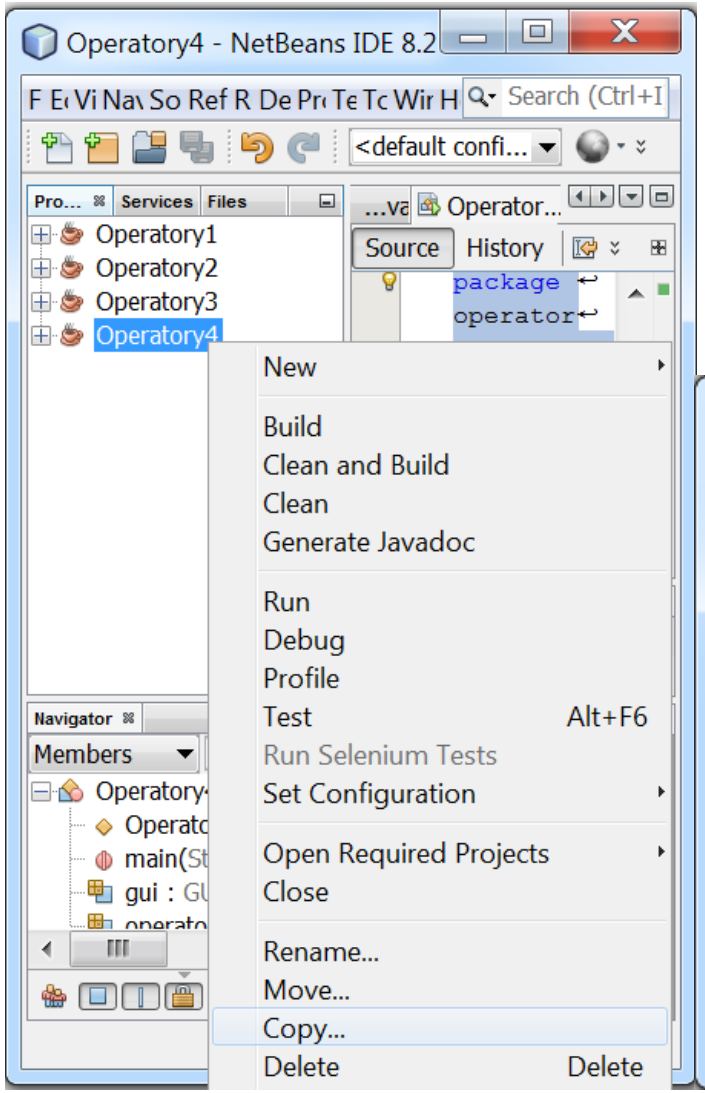

Message
arg11+arg12 wynosi 19
arg11-arg12 wynosi -21
arg11/arg12 wynosi 0
arg11*arg12 wynosi -20
arg11%arg12 wynosi -1
OK

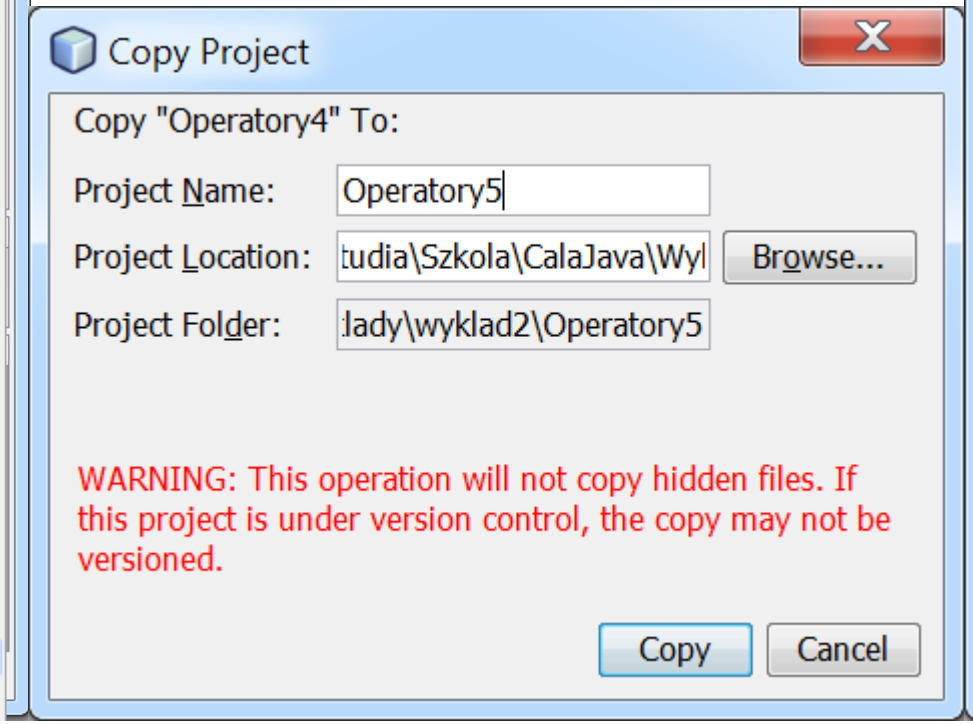

Message
arg21+arg22 wynosi 20.8
arg21-arg22 wynosi 4.2
arg21/arg22 wynosi 1.506024
arg21*arg22 wynosi 103.75
OK

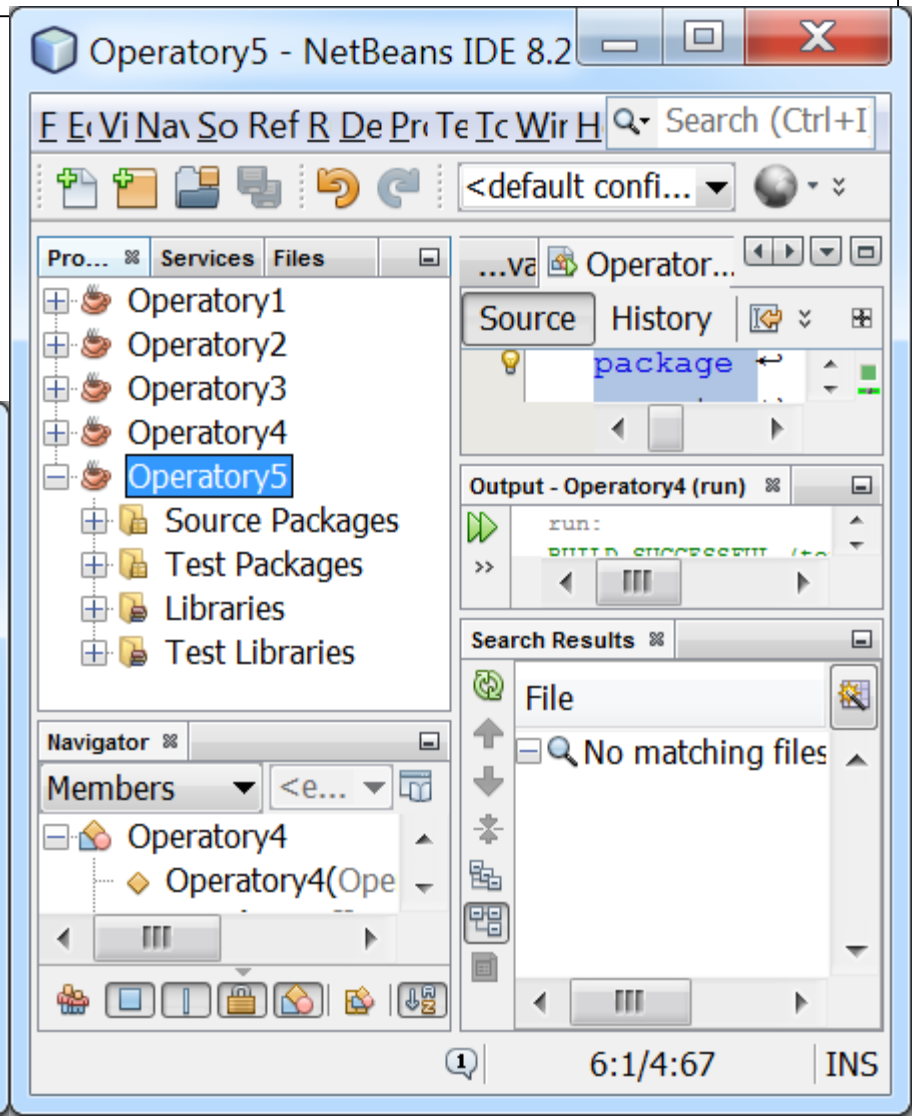

Message
wynik2 += arg21 wynosi 116.25
wynik2 -= arg21 wynosi 103.75
wynik2 *= arg21 wynosi 1296.875
wynik2 /= arg21 wynosi 103.75
OK

5. Klasa zarządzająca – składowe typu static jako atrybuty klasowe. Dostęp do składowych statycznych nie wymaga tworzenia obiektu-właściciela tych składowych.

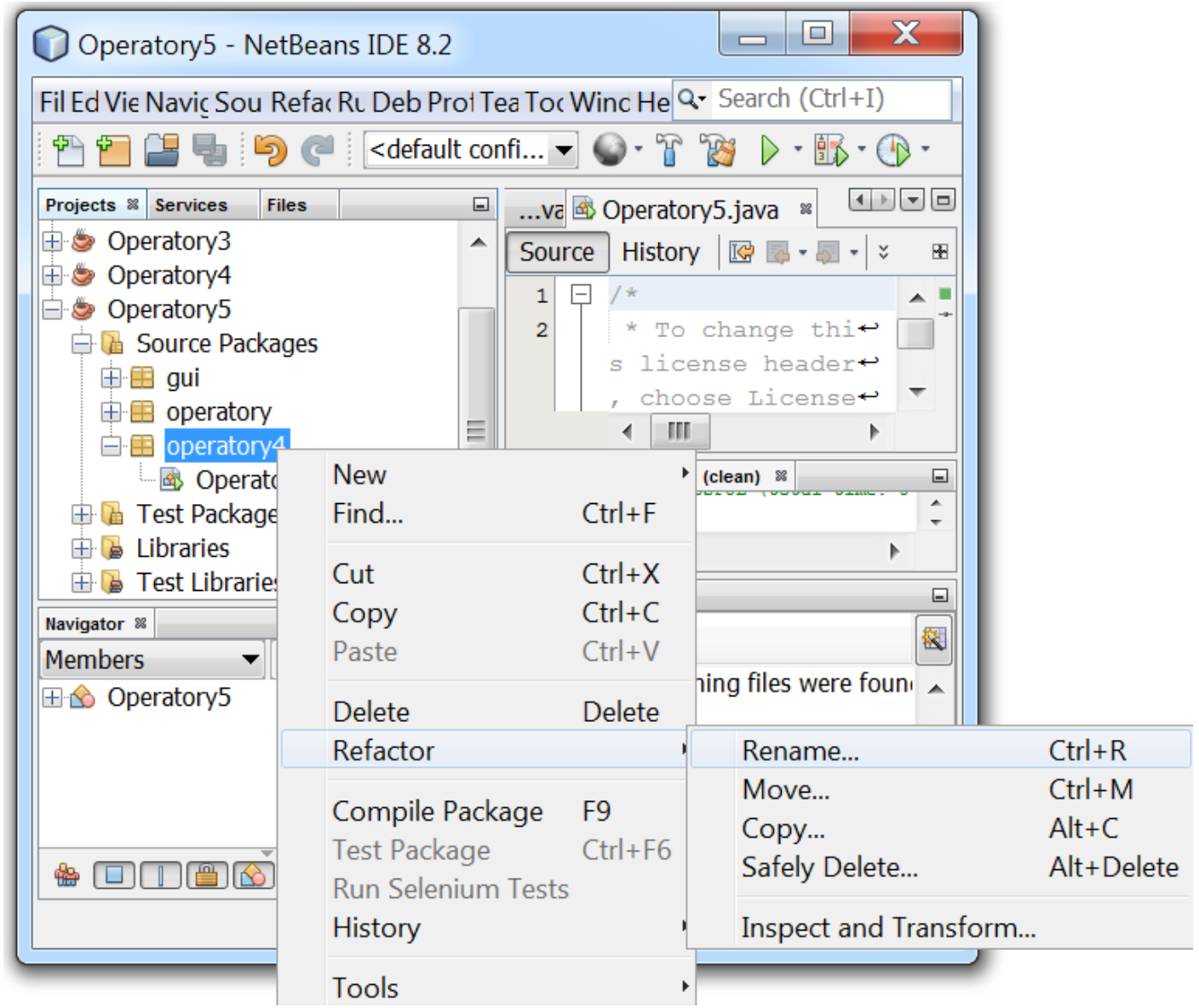
5.1. Wykonanie kopii programu Operatory4 jako projektu Operatory5

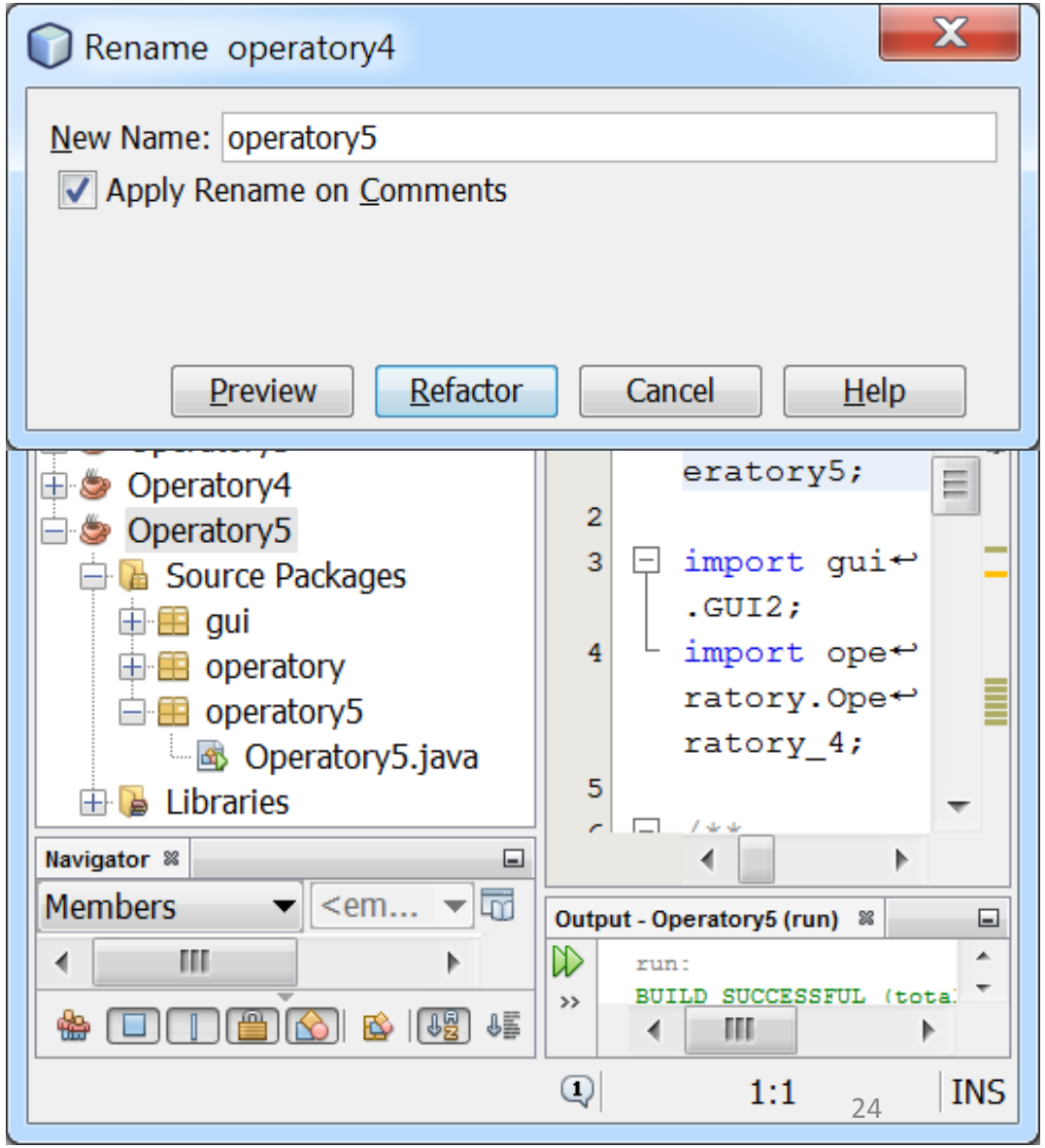






5.2. Zmiana nazwy pakietu operatory4 na operatory5 oraz nazwy klasy Operatory4 na Operatory5: nazwa/Refactor/Rename.





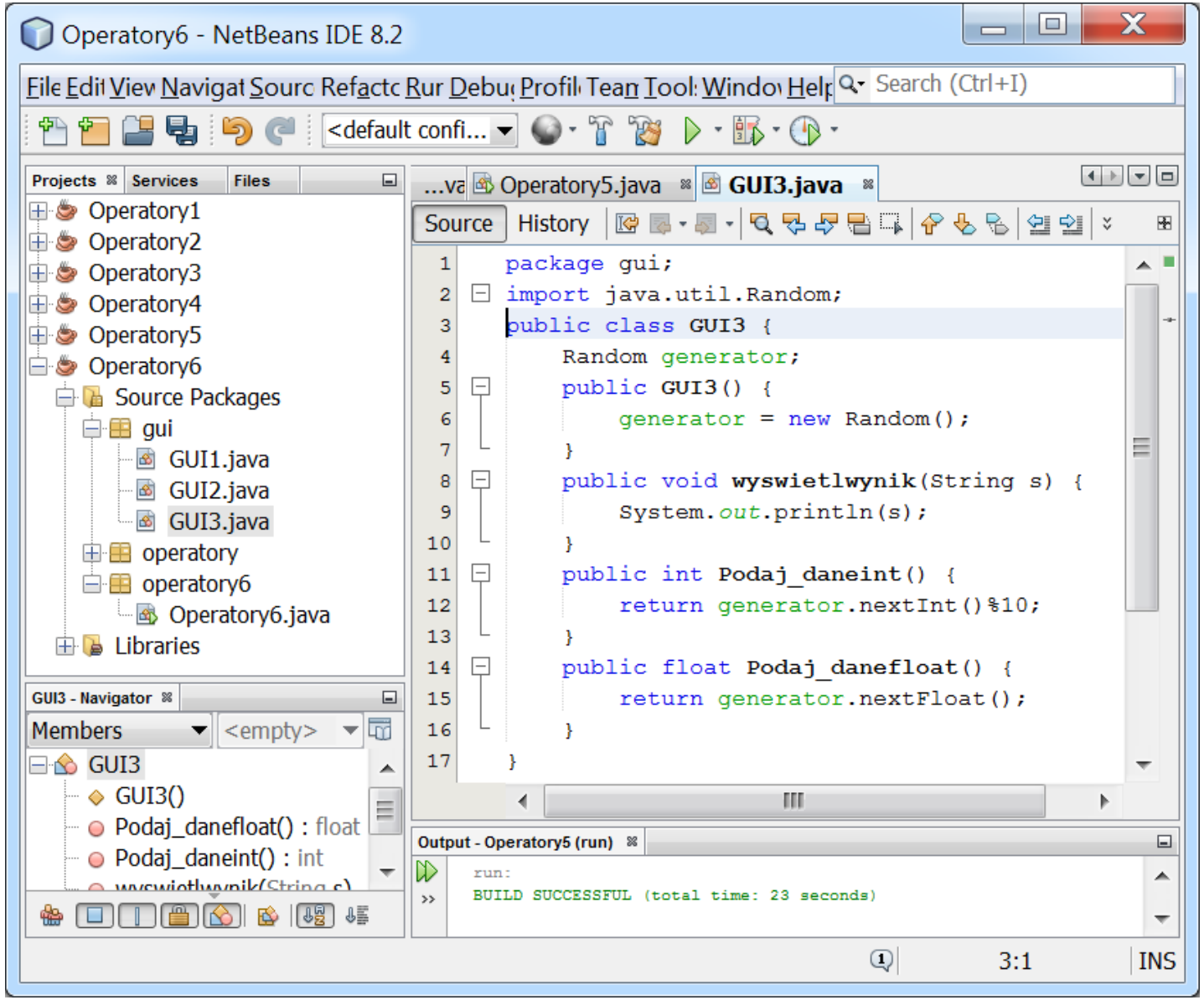
5.3. Zdefiniowanie klasy zarządzającej Operatory5 w pakiecie operatory5, utworzonych podczas tworzenia projektu oraz uruchomienie programu (wynik taki jak na slajdzie 22)

```
package operatory5;
import gui.GUI2;
import operatory.Operatory_4;

public class Operatory5 {
   static Operatory_4 operatory;
   static GUI2 gui;
   public Operatory5(Operatory_4 op, GUI2 gui) {
      operatory = op;
      this.gui = gui;
   }
   public static void main(String[] args) {
      Operatory5 op = new Operatory5(new Operatory_4(), new GUI2());
      operatory.Przypisz_argumentyint(gui.Podaj_daneint(), gui.Podaj_daneint());
      operatory.Przypisz_argumentyfloat(gui.Podaj_danefloat(), gui.Podaj_danefloat());
      gui.wyswietlwynik(operatory.dzialaniajednoarg());
      gui.wyswietlwynik(operatory.dzialaniarelacyjne());
      gui.wyswietlwynik(operatory.dzialaniaint());
      gui.wyswietlwynik(operatory.dzialaniafloat());
      gui.wyswietlwynik(operatory.dzialaniaprzypisania());  }
}
```

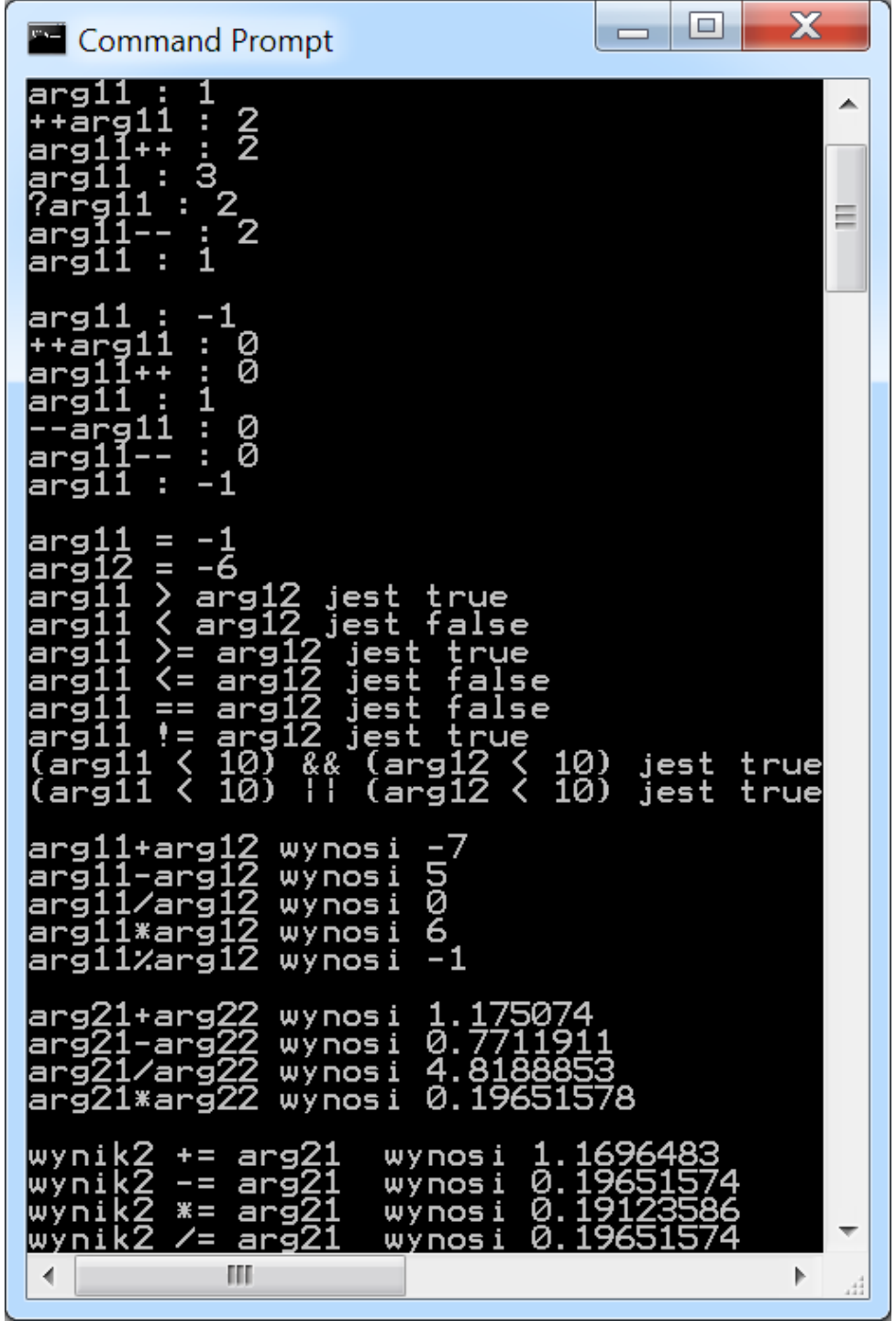
6. Wprowadzenie nowej definicji klasy GUI z zachowaniem identycznych nagłówków metod – automatyczne generowanie danych

(wykonano kopię projektu Operatory5 jako Operatory6, dodano nowa klasę GUI3 w pakiecie gui, zmieniono nazwę pakietu operatory5 jako operatory6, nazwę klasy zarządzającej Operatory5 na Operatory6 oraz zmodyfikowano jej kod)

```
package gui;

import java.util.Random;

public class GUI3 {
    Random generator;

    public GUI3() {
        generator = new Random();
    }
    public void wyswietlwynik(String s) {
        System.out.println(s);
    }
    public int Podaj_daneint() {
        return generator.nextInt()%10;
    }
    public float Podaj_danefloat() {
        return generator.nextFloat();
    }
}
```



```java
package operatory6;

import gui.GUI3;
import operatory.Operatory_4;
public class Operatory6 { //nowa klasa zarządzajaca

   static Operatory_4 operatory;
   static GUI3 gui;

   public Operatory6(Operatory_4 op, GUI3 gui) {
      operatory = op;
      this.gui = gui;
   }
   public static void main(String[] args) {
      Operatory6 op = new Operatory5(new Operatory_4(),new GUI3());
      operatory.Przypisz_argumentyint(gui.Podaj_daneint(),
                                      gui.Podaj_daneint());
      operatory.Przypisz_argumentyfloat(gui.Podaj_danefloat(),
                                        gui.Podaj_danefloat());
      gui.wyswietlwynik(operatory.dzialaniajednoarg());
      gui.wyswietlwynik(operatory.dzialaniarelacyjne());
      gui.wyswietlwynik(operatory.dzialaniaint());
      gui.wyswietlwynik(operatory.dzialaniafloat());
      gui.wyswietlwynik(operatory.dzialaniaprzypisania());
   }
}
```



7. Operatory bitowe dodane do programu Operatory6

(wykonano kopię projektu Operatory6 jako Operatory7, dodano nowa klasę Operatory_5 w pakiecie operatory, zmieniono nazwę pakietu operatory6 jako operatory7, nazwę klasy zarządzającej Operatory6 na Operatory7 oraz zmodyfikowano jej kod)

```
package operatory;

public class Operatory_5 extends Operatory_2  {

    public String dzialaniabitowe() {
       String s;
       s="arg11: "+arg11+", arg12: "+arg12+"\n";
       s += "arg11<<2 : " + (arg11 << 2) + "\n";
       s += "arg11>>2 : " + (arg11 >> 2) + "\n";

       s += "arg11>>>1 : " + (arg11 >>> 1) + "\n";
       s += "~arg12 : " + (~arg12) + "\n";
       s += "arg11&arg12 : " + (arg11 & arg12) + "\n";
       s += "arg11|arg12 : " + (arg11 | arg12) + "\n";
       s += "arg11^0x0F : " + (arg11 ^ 0x0F) + "\n";
      return s;
   }
}
```

```
package operatory7;

import gui.GUI3;
import operatory.Operatory_5;

public class Operatory7 {       //nowa klasa zarządzajaca

   static Operatory_5 operatory;
   static GUI3 gui;

   public Operatory7(Operatory_5 op, GUI3 gui_) {
      operatory = op;
      gui = gui_;
   }

  public static void main(String[] args) {
      Operatory7 op = new Operatory7(new Operatory_5(), new GUI3());
      operatory.Przypisz_argumentyint(gui.Podaj_daneint(), gui.Podaj_daneint());
                        //gui.wyswietlwynik(operatory.dzialaniajednoarg()); - wynik, jak w programie Operatory2, slajd 15
                        //gui.wyswietlwynik(operatory.dzialaniarelacyjne()); - wynik, jak w programie Operatory2, slajd 15
      gui.wyswietlwynik(operatory.dzialaniabitowe());
   }
}
```

```
run:
arg11: 4, arg12: -7
arg11<<2 : 16
arg11>>2 : 1
arg11>>>1 : 2
~arg12 : 6
arg11&arg12 : 0
arg11|arg12 : -3
arg11^0x0F : 11

BUILD SUCCESSFUL (total time: 0 seconds)
```

8. Podsumowanie – pakiety
DOSTĘP DO METOD I ZMIENNYCH

Dostęp do zmiennych (na podstawie:L.Lemay,R.Cadenhead, Java 2 dla każdego, Helion 2001)

Zakres „widzialności” metody lub zmiennej	public	protected	**domyślny (brak słowa kluczowego)**	private
Klasa	tak	tak	tak	tak
Pakiet	tak	tak	tak	nie
Inny pakiet	tak	nie	nie	nie
Podklasa, **pakiet**	tak	tak	tak	nie
Podklasa, inny pakiet	tak	tak	nie	nie