

Komunikatory typu TCP/IP

lab2

Dr inż. Zofia Kruczkiewicz

Programowanie aplikacji
internetowych

Zadanie1 - klient wysyła jeden komunikat (typu String) do serwera i kończy swoje istnienie, a serwer go odbiera

Uruchom program **Nowykomunikator1**, w którym **klient wysyła jeden komunikat do serwera i kończy swoje istnienie, a serwer go odbiera**. Dokonaj analizy programu (sposób uruchomienia komunikatora na następnym slajdzie)

1. Wyjaśnij rolę konstruktora i metody **run** w klasie **nowyserwer1**.
2. Wyjaśnij poszczególne wywołania w metodzie **run** **nowyserwer1**:

```
gniazdo_klienta = serwer.accept();
```

oraz

```
wyjście = new ObjectOutputStream(gniazdo_klienta.getOutputStream());
```

```
wyjście.flush();
```

```
wejście = new ObjectInputStream(gniazdo_klienta.getInputStream());
```

```
komponent_nowegoklienta1 komp_klienta =
```

```
new komponent_nowegoklienta1(gniazdo_klienta, wejście, wyjście);
```

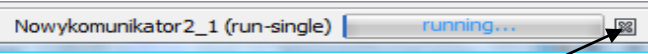
```
Thread watek_komponentu_klienta = new Thread(komp_klienta);
```

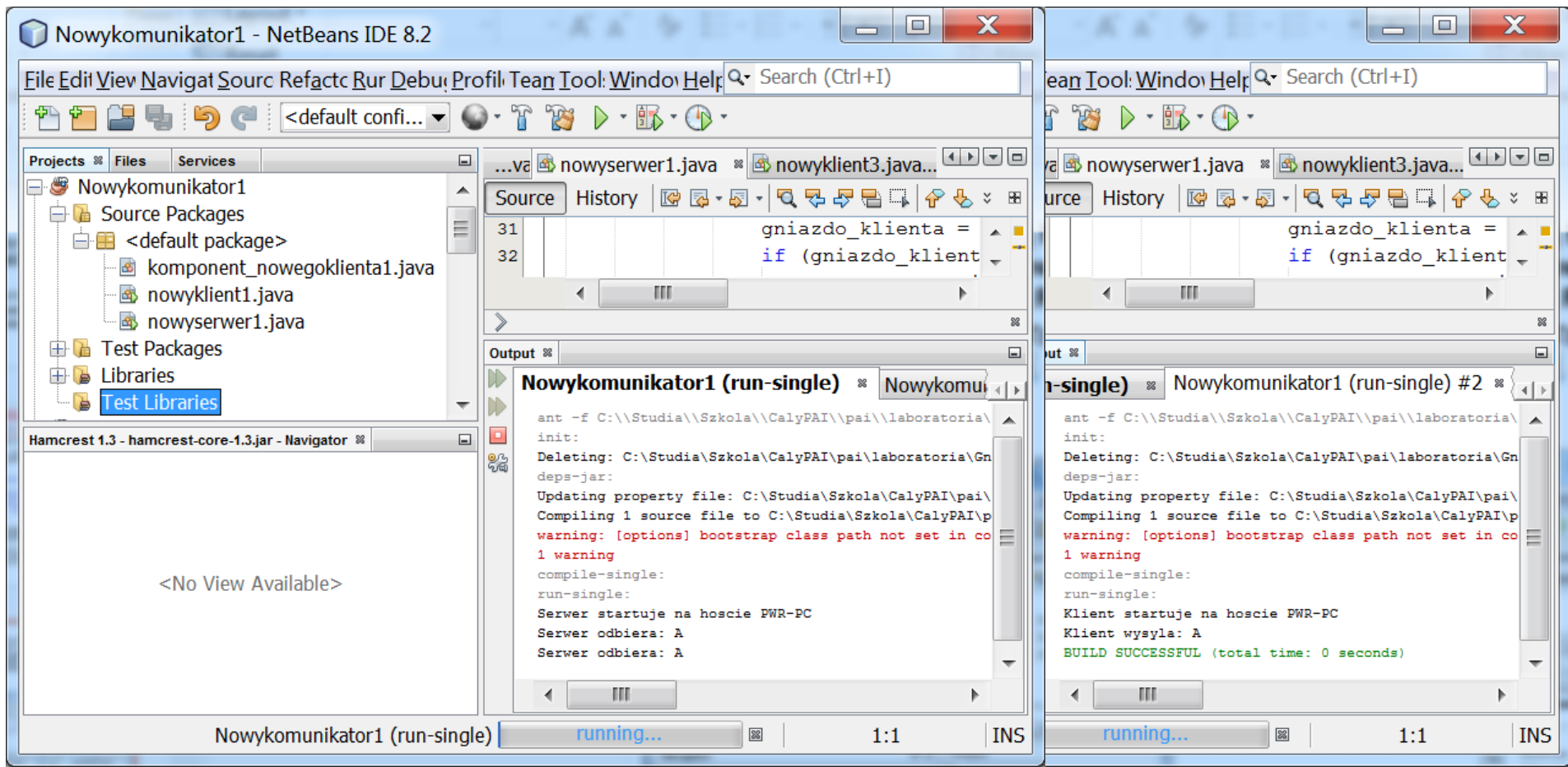
```
watek_komponentu_klienta.start();
```

3. Wyjaśnij rolę klasy **nowy klient1** oraz zawartość **konstruktora** i metody **run** tej klasy (realizacja protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **OutputObjectStream** oraz **InputObjectStream**)
4. Wyjaśnij rolę klasy **komponent_nowegoklienta1** oraz zawartość metody **run** tej klasy (realizacja protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **OutputObjectStream** oraz **InputObjectStream**)

5. Uruchomienie:

- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer1** i wybrać opcję **Run File**)
- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowy klient1** i wybrać opcję **Run File**)

-  Aby zakończyć program serwera, należy kliknąć na wskazany znak.



The image displays two side-by-side screenshots of the NetBeans IDE 8.2 interface, illustrating the execution of a Java application.

Left Screenshot: Shows the 'Projects' view on the left, with 'nowyserwer1.java' selected in the source editor. The 'Output' window displays the execution log for 'Nowykomunikator1 (run-single)'. The log shows the following output:

```
ant -f C:\Studia\Szkola\CalyPAI\pai\laboratoria\
init:
Deleting: C:\Studia\Szkola\CalyPAI\pai\laboratoria\Gn
deps-jar:
Updating property file: C:\Studia\Szkola\CalyPAI\pai\
Compiling 1 source file to C:\Studia\Szkola\CalyPAI\p
warning: [options] bootstrap class path not set in co
1 warning
compile-single:
run-single:
Serwer startuje na hoscie PWR-PC
Serwer odbiera: A
Serwer odbiera: A
```

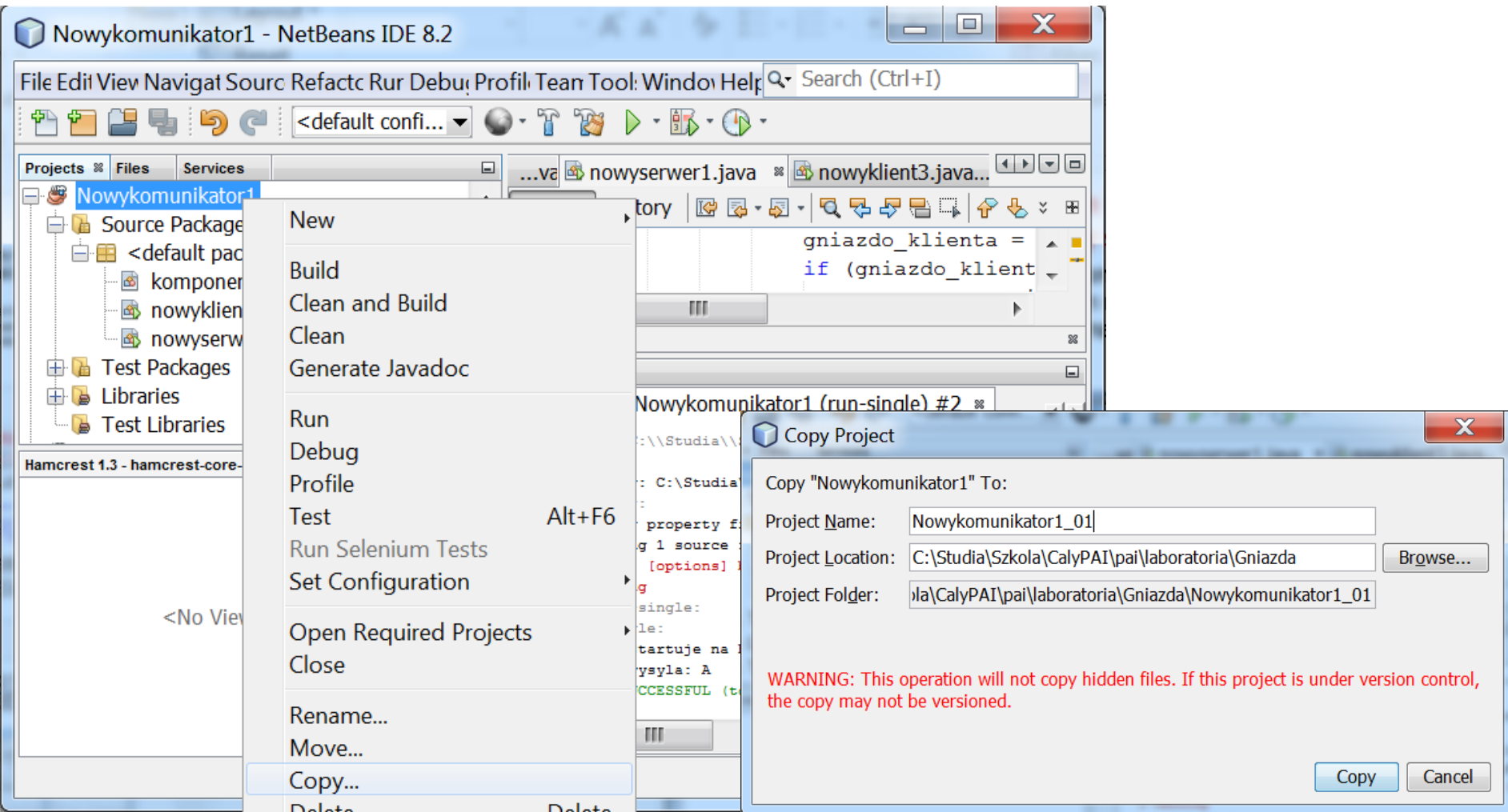
Right Screenshot: Shows the 'Output' window with the execution log for 'Nowykomunikator1 (run-single) #2'. The log shows the following output:

```
ant -f C:\Studia\Szkola\CalyPAI\pai\laboratoria\
init:
Deleting: C:\Studia\Szkola\CalyPAI\pai\laboratoria\Gn
deps-jar:
Updating property file: C:\Studia\Szkola\CalyPAI\pai\
Compiling 1 source file to C:\Studia\Szkola\CalyPAI\p
warning: [options] bootstrap class path not set in co
1 warning
compile-single:
run-single:
Klient startuje na hoscie PWR-PC
Klient wysyla: A
BUILD SUCCESSFUL (total time: 0 seconds)
```

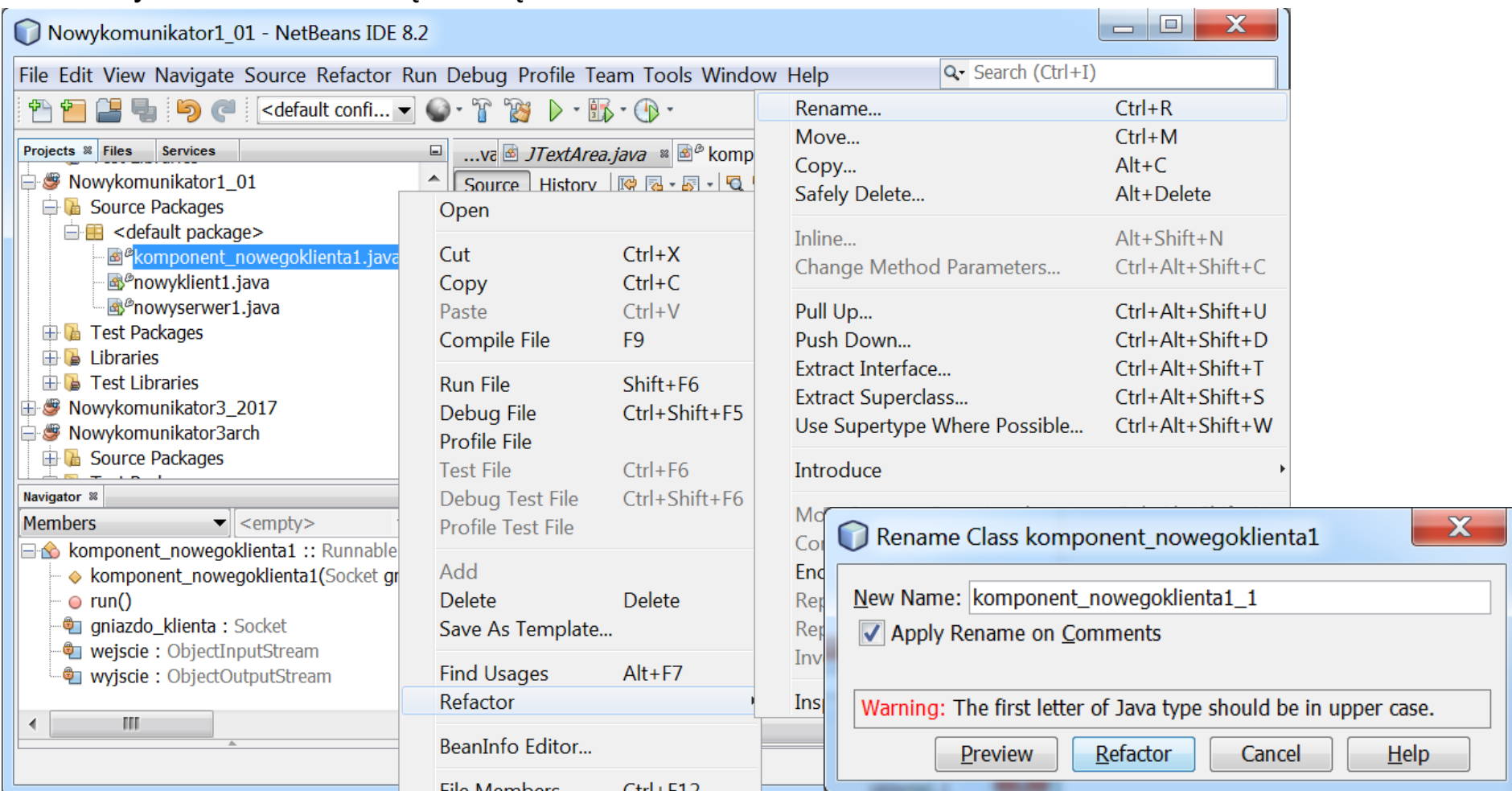
Both screenshots show the status bar at the bottom indicating the application is running: 'Nowykomunikator1 (run-single) running... 1:1 INS'.

Zadanie2 - klient wysła jeden komunikat (typ zdefiniowany przez programistę) do serwera i kończy swoje istnienie, a serwer go odbiera

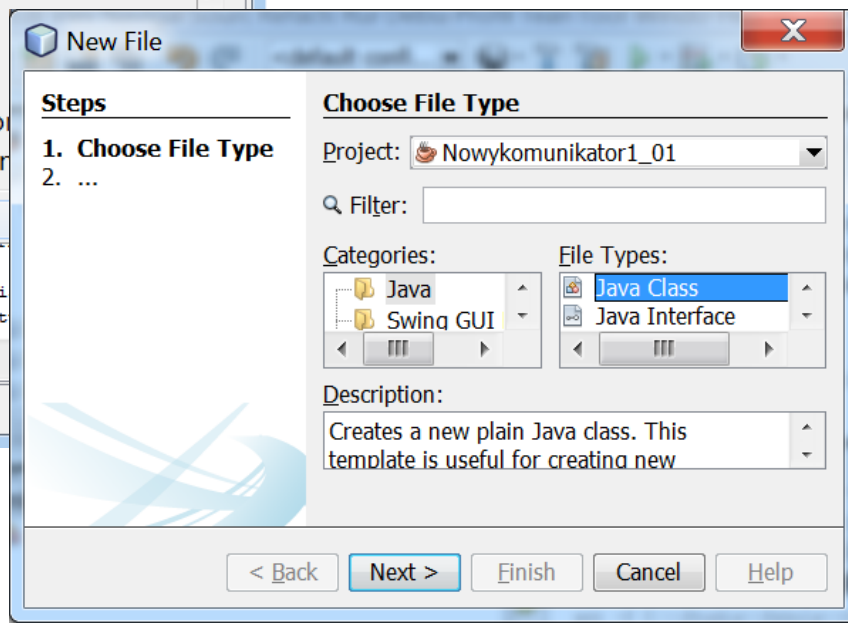
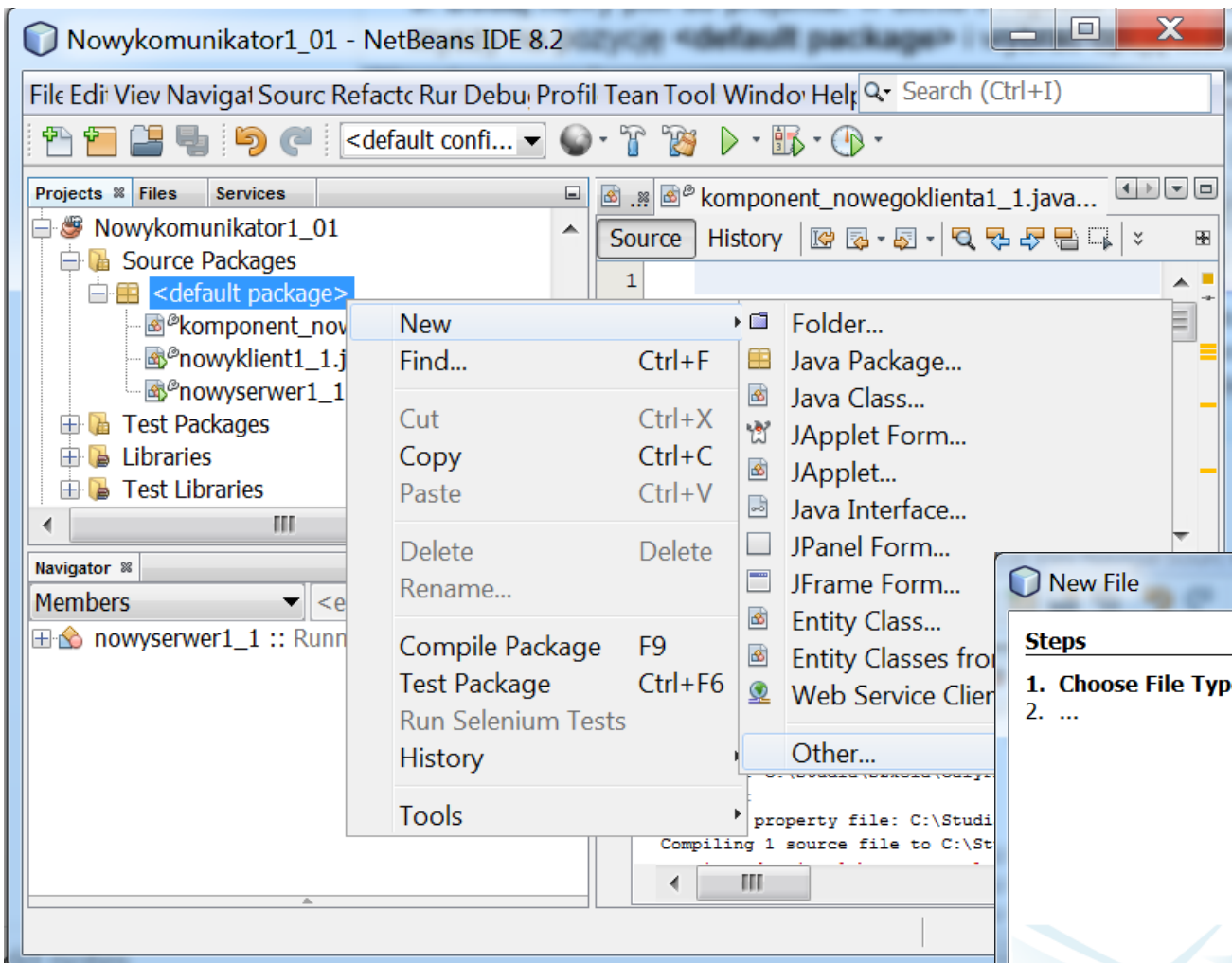
- Wykonaj kopię programu Nowykomunikator1 jako Nowykomunikator1_01 - w oknie **Projects** należy kliknąć prawym klawiszem myszy na projekt **Nowykomunikator1** i wybrać opcję **Copy**. W okienku **Copy Projekt** podać nazwę nowego projektu (Project Name) oraz położenie (Project Location).



2. Należy również zmienić nazwy wszystkich plików projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pliki projektu **Nowykomunikator1_1** i wybrać opcję **Refactor<Rename**. W oknie **Rename Class** zaznaczając **Apply Rename on Comments** podać odpowiednio nowe nawy klas:
- dla nowy klient1 podać nowy klient1_1
 - dla nowy serwer1 podać nowy serwer1_1
 - dla komponent_nowego klienta1 podać komponent_nowego klienta1_1
- i należy zatwierdzić nową nazwę klawiszem **Refactor**.



3. Dodaj nowy plik do projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **<default package>** i wybrać opcję **New/Other/Java/Java Class**



W okienku **New Java Class** nadac nowej klasie nazwę **komunikat**

New Java Class

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

Class Name:

Project:

Location:

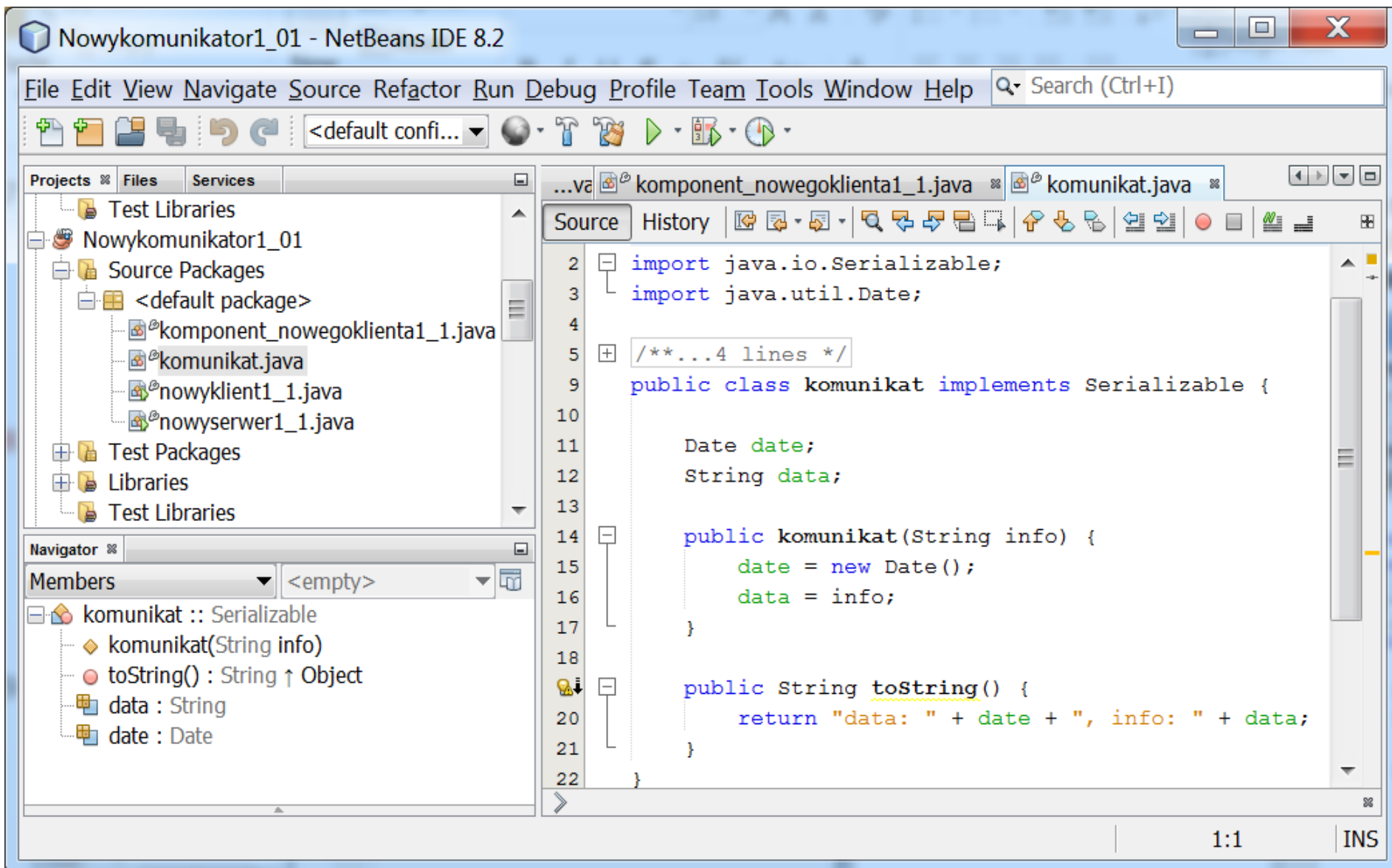
Package:

Created File:

Warning: It is highly recommended that you do not place Java class

< Back Next > **Finish** Cancel Help

W okienku edytora pliku **komunikat.java** należy dodać podany poniżej kod. Należy zwrócić uwagę na konieczność serializacji nowej klasy, która powinna zastąpić klasę **String**, używaną jako typ przesyłanych danych między klientem i serwerem.

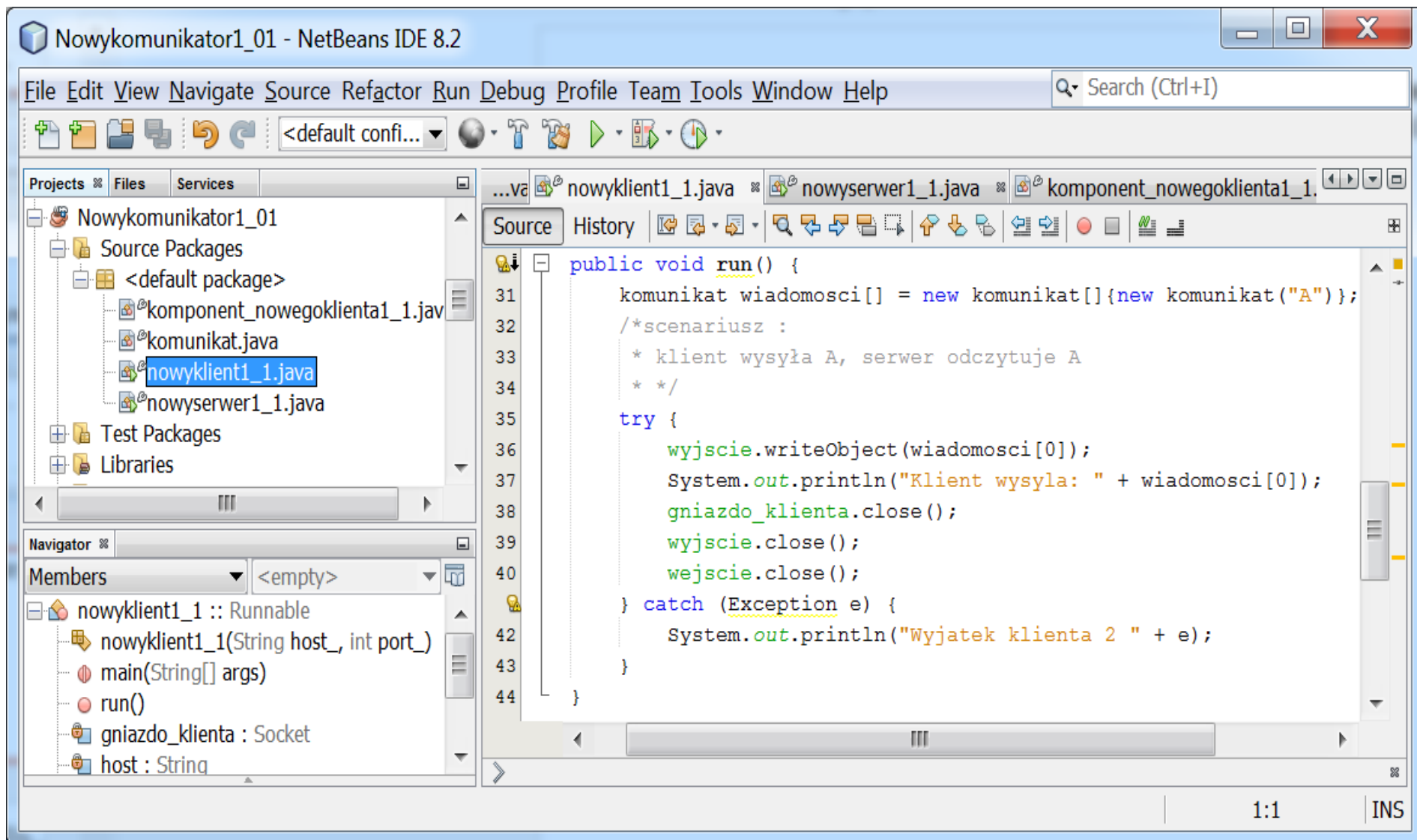


The screenshot shows the NetBeans IDE 8.2 interface. The main editor window displays the source code for `komunikat.java`. The code is as follows:

```
2 import java.io.Serializable;
3 import java.util.Date;
4
5 /**...4 lines */
9 public class komunikat implements Serializable {
10
11     Date date;
12     String data;
13
14     public komunikat(String info) {
15         date = new Date();
16         data = info;
17     }
18
19     public String toString() {
20         return "data: " + date + ", info: " + data;
21     }
22 }
```

The left sidebar shows the project structure for `Nowykomunikator1_01`, including source packages and test packages. The `komunikat.java` file is selected in the `Source Packages` view. The `Navigator` window at the bottom left shows the class `komunikat` implementing `Serializable`, with its methods `komunikat(String info)` and `toString() : String`, and its attributes `data : String` and `date : Date`.

4. Oto zmiana, jaka należy wykonać w kodzie w kodzie klienta klasy **nowy klient1_1**. Należy odpowiednio zmienić kod metody **run** klasy **komponent_nowegoklienta1_1**, zmieniając typ odbieranego komunikatu z typu **String** na typ **komunikat**



The screenshot shows the NetBeans IDE 8.2 interface. The main editor displays the code for the `run()` method in the `KomponentNowegoklienta1_1` class. The code is as follows:

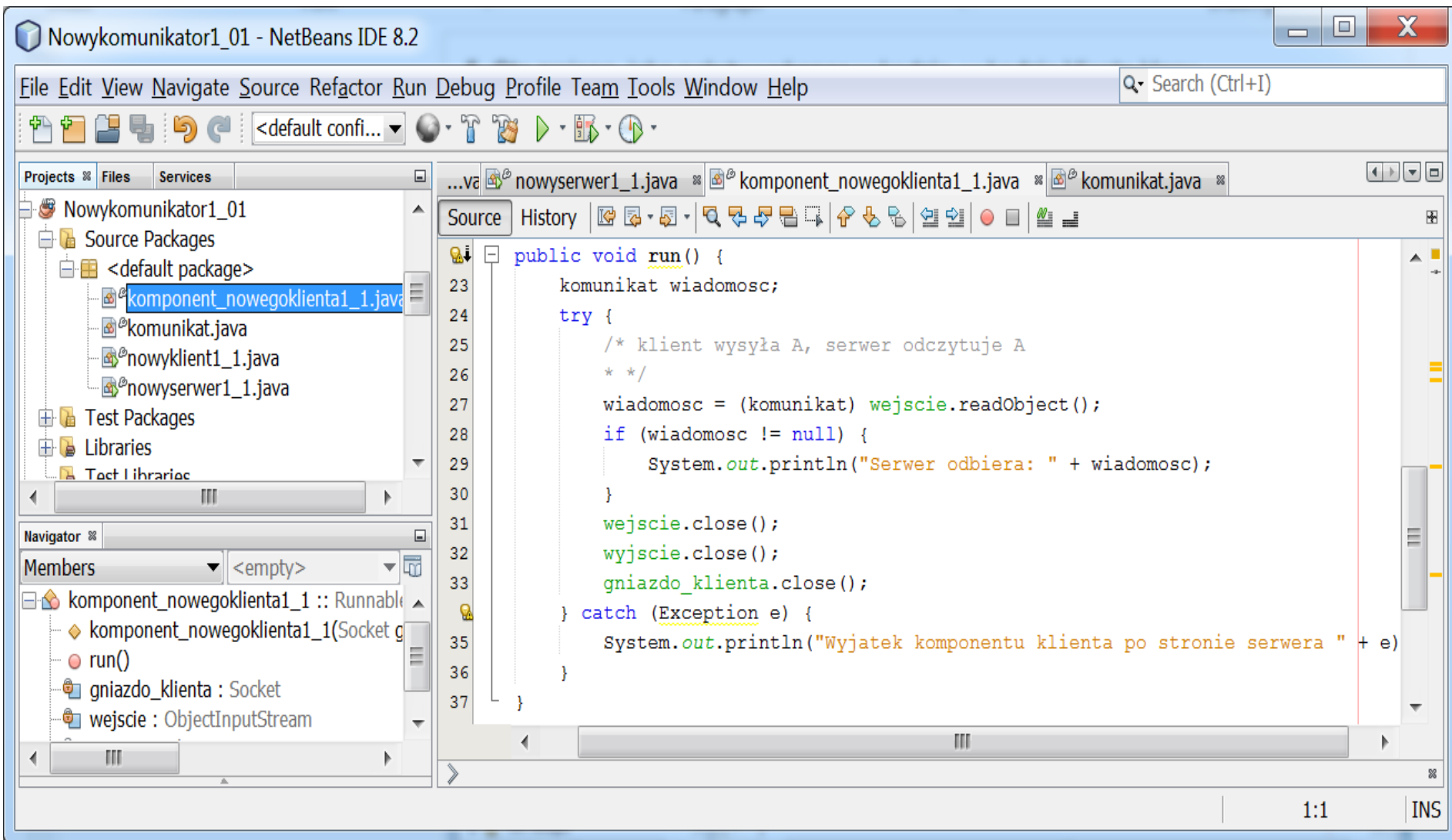
```
public void run() {
    komunikat wiadomosci[] = new komunikat[]{new komunikat("A")};
    /*scenariusz :
    * klient wysyła A, serwer odczytuje A
    * */
    try {
        wyjście.writeObject(wiadomosci[0]);
        System.out.println("Klient wysyła: " + wiadomosci[0]);
        gniazdo_klienta.close();
        wyjście.close();
        wejście.close();
    } catch (Exception e) {
        System.out.println("Wyjatek klienta 2 " + e);
    }
}
```

The left sidebar shows the project structure for `Nowykomunikator1_01`, with `nowy klient1_1.java` selected. The bottom status bar shows the zoom level as 1:1 and the cursor position as INS.

5. Oto zmiana, jaka należy wykonać w kodzie w kodzie klienta klasy

komponent_nowegoklienta1_1.

Należy odpowiednio zmienić kod metody **run** klasy **komponent_nowegoklienta1_1**, zmieniając typ odbieranego komunikatu **wiadomosc** z typu **String** na typ **komunikat**.



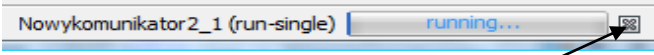
The screenshot shows the NetBeans IDE 8.2 interface. The main editor window displays the code for the `run()` method in the `komponent_nowegoklienta1_1.java` file. The code is as follows:

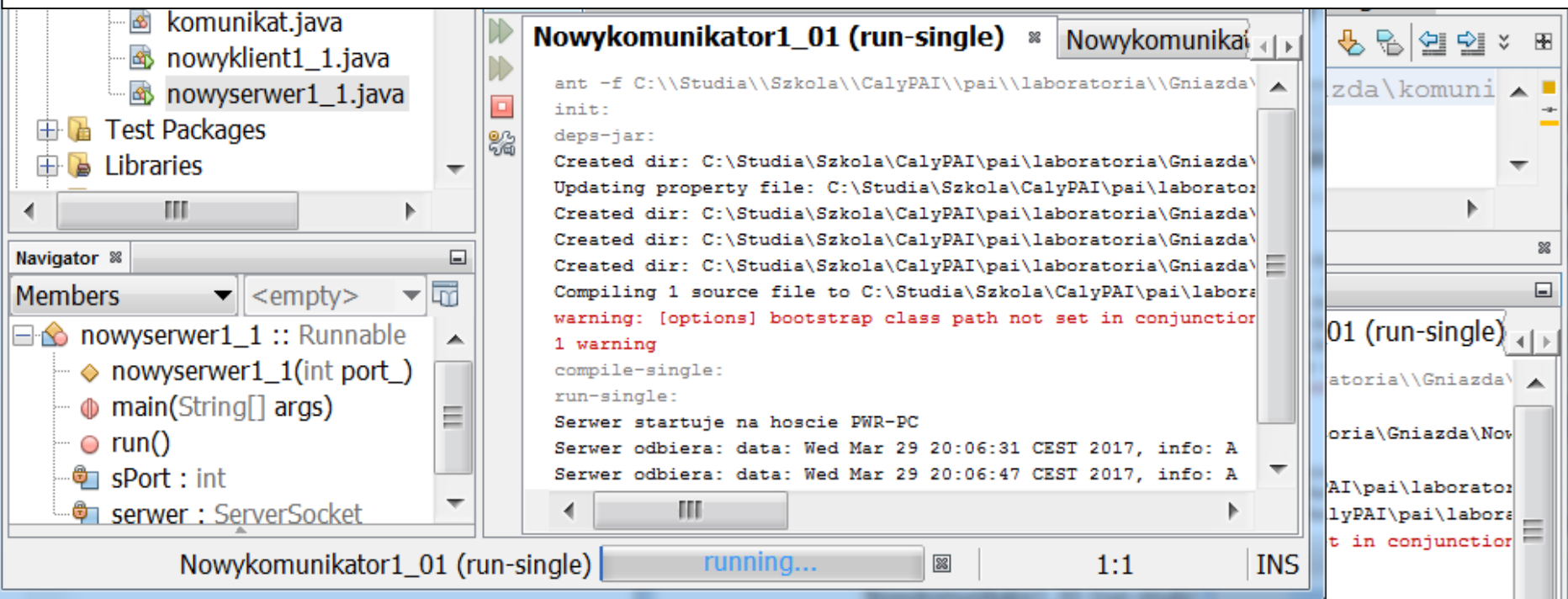
```
public void run() {
    komunikat wiadomosc;
    try {
        /* klient wysła A, serwer odczytuje A
        */
        wiadomosc = (komunikat) wejscie.readObject();
        if (wiadomosc != null) {
            System.out.println("Serwer odbiera: " + wiadomosc);
        }
        wejscie.close();
        wyjscie.close();
        gniazdo_klienta.close();
    } catch (Exception e) {
        System.out.println("Wyjatek komponentu klienta po stronie serwera " + e);
    }
}
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar, and a project explorer on the left showing the project structure for "Nowykomunikator1_01". The Navigator window at the bottom left shows the members of the `komponent_nowegoklienta1_1` class, including `run()`, `gniazdo_klienta`, and `wejscie`.

6. Uruchomienie:

- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer1_1** i wybrać opcję **Run File**)
- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowy klient1_1** i wybrać opcję **Run File**)

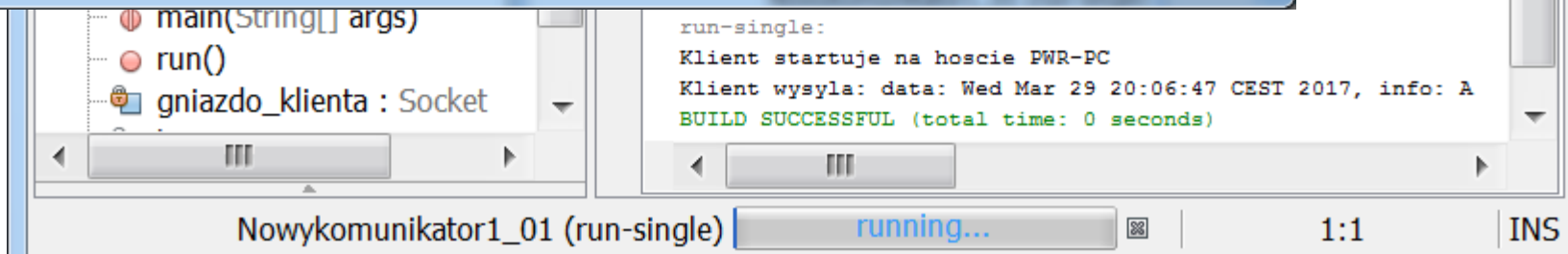
 - Aby zakończyć program serwera, należy kliknąć na wskazany znak.



The screenshot shows the IDE interface with the following components:

- Project Explorer:** Shows files `komunikat.java`, `nowy klient1_1.java`, and `nowyserwer1_1.java`.
- Navigator:** Shows the class `nowyserwer1_1 :: Runnable` with methods `nowyserwer1_1(int port_)`, `main(String[] args)`, `run()`, `sPort : int`, and `serwer : ServerSocket`.
- Console:** Shows the output for `Nowykomunikator1_01 (run-single)`. The output includes:

```
ant -f C:\\Studia\\Szkola\\CalyPAI\\pai\\laboratoria\\Gniazda\\
init:
deps-jar:
Created dir: C:\\Studia\\Szkola\\CalyPAI\\pai\\laboratoria\\Gniazda\\
Updating property file: C:\\Studia\\Szkola\\CalyPAI\\pai\\laborator
Created dir: C:\\Studia\\Szkola\\CalyPAI\\pai\\laboratoria\\Gniazda\\
Created dir: C:\\Studia\\Szkola\\CalyPAI\\pai\\laboratoria\\Gniazda\\
Created dir: C:\\Studia\\Szkola\\CalyPAI\\pai\\laboratoria\\Gniazda\\
Compiling 1 source file to C:\\Studia\\Szkola\\CalyPAI\\pai\\labore
warning: [options] bootstrap class path not set in conjunction
1 warning
compile-single:
run-single:
Serwer startuje na hoscie PWR-PC
Serwer odbiera: data: Wed Mar 29 20:06:31 CEST 2017, info: A
Serwer odbiera: data: Wed Mar 29 20:06:47 CEST 2017, info: A
```
- Status Bar:** Shows `Nowykomunikator1_01 (run-single) | running... | 1:1 | INS`.



The screenshot shows the IDE interface with the following components:

- Project Explorer:** Shows files `main(String[] args)`, `run()`, and `gniazdo_klienta : Socket`.
- Console:** Shows the output for `run-single`. The output includes:

```
run-single:
Klient startuje na hoscie PWR-PC
Klient wysyla: data: Wed Mar 29 20:06:47 CEST 2017, info: A
BUILD SUCCESSFUL (total time: 0 seconds)
```
- Status Bar:** Shows `Nowykomunikator1_01 (run-single) | running... | 1:1 | INS`.

Zadanie 3 – typ komunikatu - String

/*protokół:

- * klient wysła A, serwer odczytuje A**
- * serwer wysła AA, klient odczytuje AA**
- * klient wysła B, serwer odczytuje B**
- * serwer wysła BB, klient odczytuje BB**
- * klient wysła C, serwer odczytuje C**
- * serwer wysła CC, klient odczytuje CC**
- * */**

1. Wykonaj program **Nowykomunikator2**, który realizuje protokół wymiany komunikatów, podany na kolejnym slajdzie, jako kopię projektu **Nowykomunikator1** - w oknie **Projects** należy kliknąć prawym klawiszem myszy na projekt **Nowykomunikator1** i wybrać opcję **Copy**. W okienku **Copy Projekt** podać nazwę nowego projektu (Project Name) oraz położenie (Project Location).
2. Należy również zmienić nazwy wszystkich plików projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pliki projektu **Nowykomunikator2** i wybrać opcję **Refactor<Rename**. W oknie **Rename Class** zaznaczając **Apply Rename on Comments** podać odpowiednio nowe nazwy klas:
 - dla nowy klient1 podać nowy klient2
 - dla nowy serwer1 podać nowy serwer2
 - dla komponent_nowego klienta1 podać komponent_nowego klienta2i należy zatwierdzić nową nazwę klawiszem **Refactor**.

3. Zmiana kodu klasy klienta **nowy klient2** – w metodzie **run** dokonano zmiany protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **OutputStream** oraz **InputStream**

The screenshot shows an IDE with the following components:

- Projects View:** Shows a project structure with packages like 'Source Packages' and 'Test Packages'. The file 'nowy klient2.java' is highlighted.
- Navigator:** Shows the class 'nowy klient2 :: Runnable' with methods 'nowy klient2(String host_, int port)', 'main(String[] args)', and 'run()'. It also lists attributes: 'gniazdo_klienta : Socket', 'host : String', 'port : int', 'wejście : ObjectInputStream', and 'wyjście : ObjectOutputStream'.
- Code Editor:** Displays the code for 'nowy klient2.java'. The 'run()' method is shown with the following code:

```
public void run() {
    String wiadomosc;
    String wiadomosci[] = {"A", "B", "C"};
    /*scenariusz :
    * klient wysyła A, serwer odczytuje A
    * serwer wysyła AA, klient odczytuje AA
    * klient wysyła B, serwer odczytuje B
    * serwer wysyła BB, klient odczytuje BB
    * klient wysyła C, serwer odczytuje C
    * serwer wysyła CC, klient odczytuje CC
    * */
    try {
        for (int i = 0; i < wiadomosci.length; i++) {
            wyjście.writeObject(wiadomosci[i]);
            System.out.println("Klient wysyła: " + wiadomosci[i]);
            wiadomosc = (String) wejście.readObject();
            if(wiadomosc!=null)
                System.out.println("Klient odbiera: " + wiadomosc);
            else break;
        }
        gniazdo_klienta.close();
        wyjście.close();
        wejście.close();
    } catch (Exception e) {
        System.out.println("Wyjatek klienta 2 " + e);
    }
}
```
- Output View:** Shows the output of the program, which is currently empty.

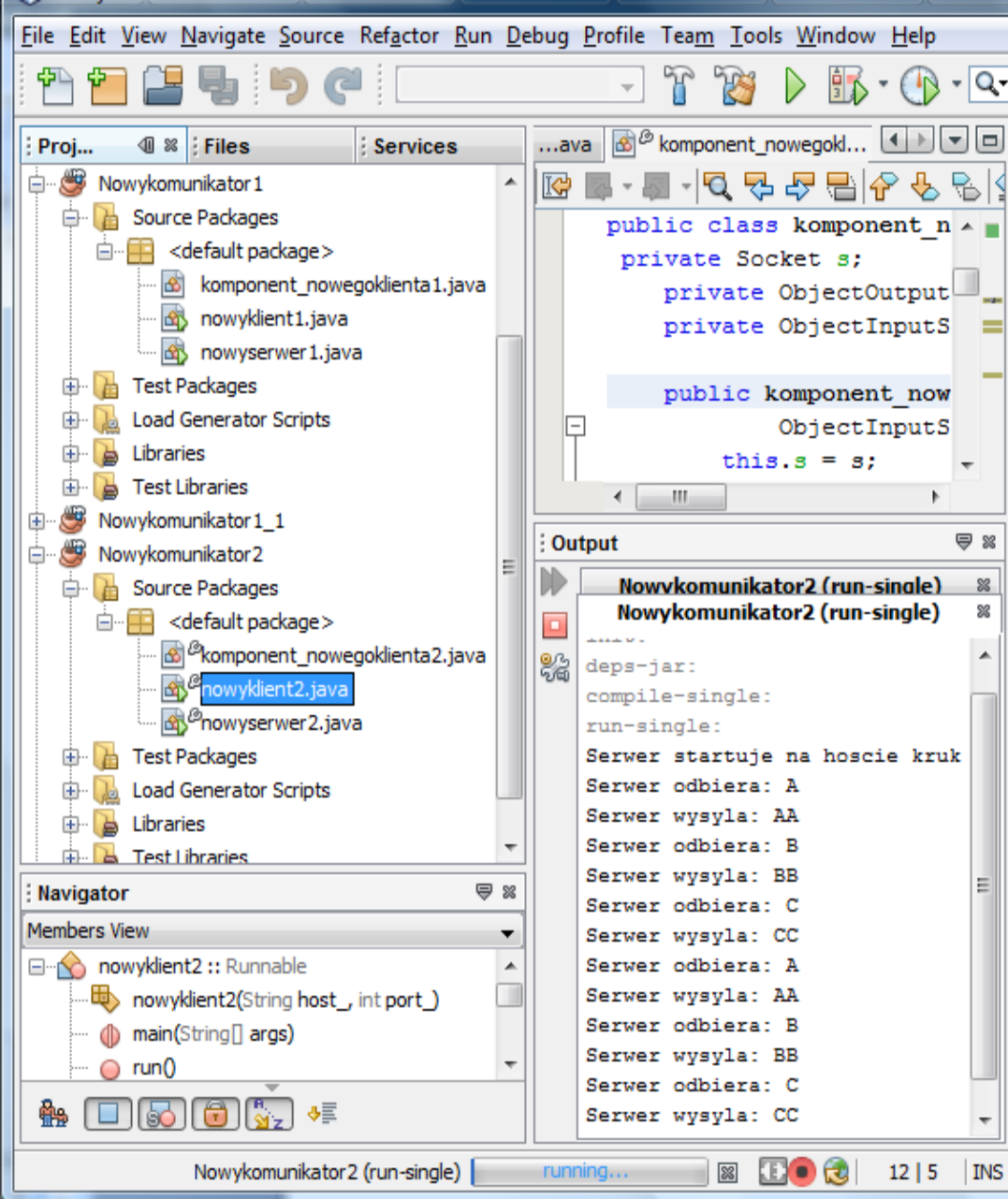
Red boxes in the code editor highlight the array initialization `String wiadomosci[] = {"A", "B", "C"};` and the `for` loop logic.

4. Zmiana kodu klasy klienta **komponent_nowegoklienta2** – w metodzie **run** dokonano zmiany protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **ObjectOutputStream** oraz **ObjectInputStream**

The screenshot shows an IDE with the following components:

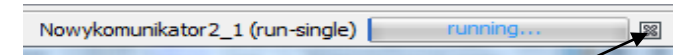
- Projects:** A tree view on the left showing a project structure with packages like `komponent_nowegokli`, `nowy klient2.java`, and `nowy serwer2.java`.
- Navigator:** A panel below the projects showing the members of the `komponent_nowegoklienta2` class, including `run()`, `gniazdo_klienta`, `wejście`, and `wyjście`.
- Source:** The main editor window showing the `run()` method in `komponent_nowegoklienta2.java`. The code is as follows:

```
public void run() {
    String wiadomosc;
    String wiadomosci[] = {"AA", "BB", "CC"};
    try {
        /* klient wysła A, serwer odczytuje A
        * serwer wysła AA, klient odczytuje AA
        * klient wysła B, serwer odczytuje B
        * serwer wysła BB, klient odczytuje BB
        * klient wysła C, serwer odczytuje C
        * serwer wysła CC, klient odczytuje CC
        */
        for (int i = 0; i < wiadomosci.length; i++) {
            wiadomosc = (String) wejście.readObject();
            if (wiadomosc != null) {
                System.out.println("Serwer odbiera: " + wiadomosc);
            } else {
                break;
            }
            wyjście.writeObject(wiadomosci[i]);
            System.out.println("Serwer wysła: " + wiadomosci[i]);
        }
        wejście.close();
        wyjście.close();
        gniazdo_klienta.close();
    } catch (Exception e) {
        System.out.println("Wyjatek komponentu klienta po stronie serwera " + e);
    }
}
```
- Output:** A panel at the bottom showing the output of the program, which is currently empty.

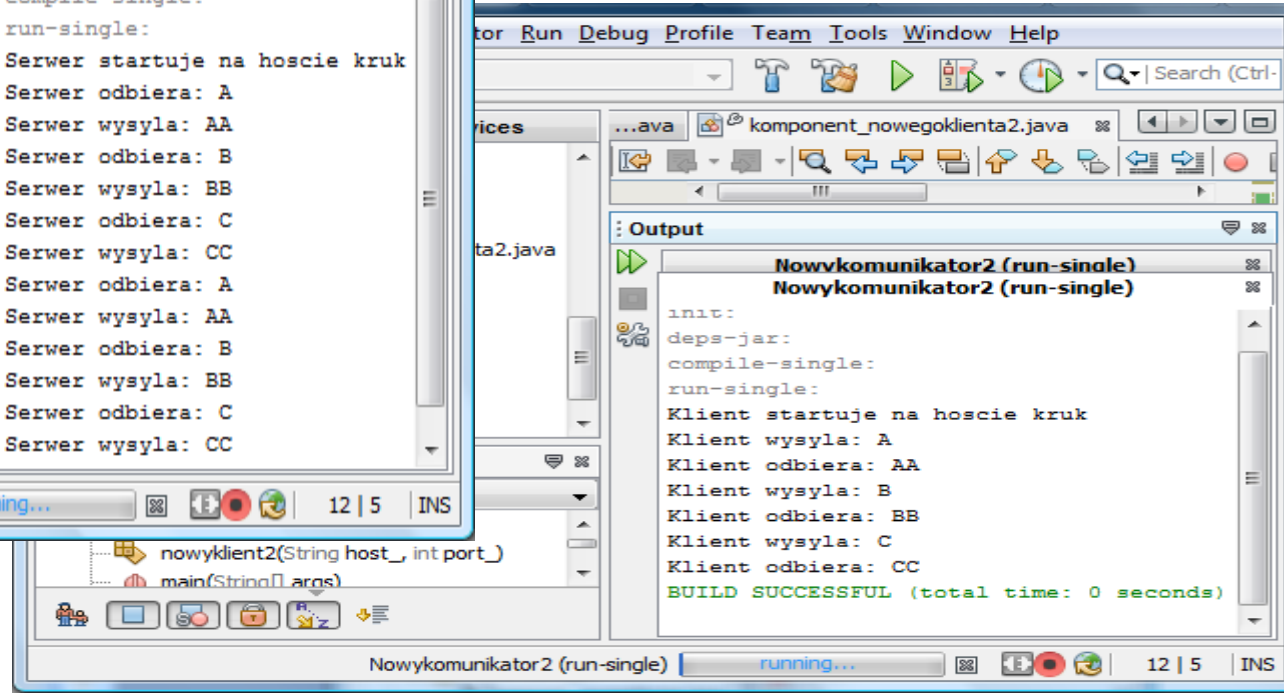


5. Uruchomienie:

- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer2** i wybrać opcję **Run File**)
- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowy klient2** i wybrać opcję **Run File**)
- Należy zakończyć wątek serwera.



Aby zakończyć ten program należy kliknąć na wskazany znak.



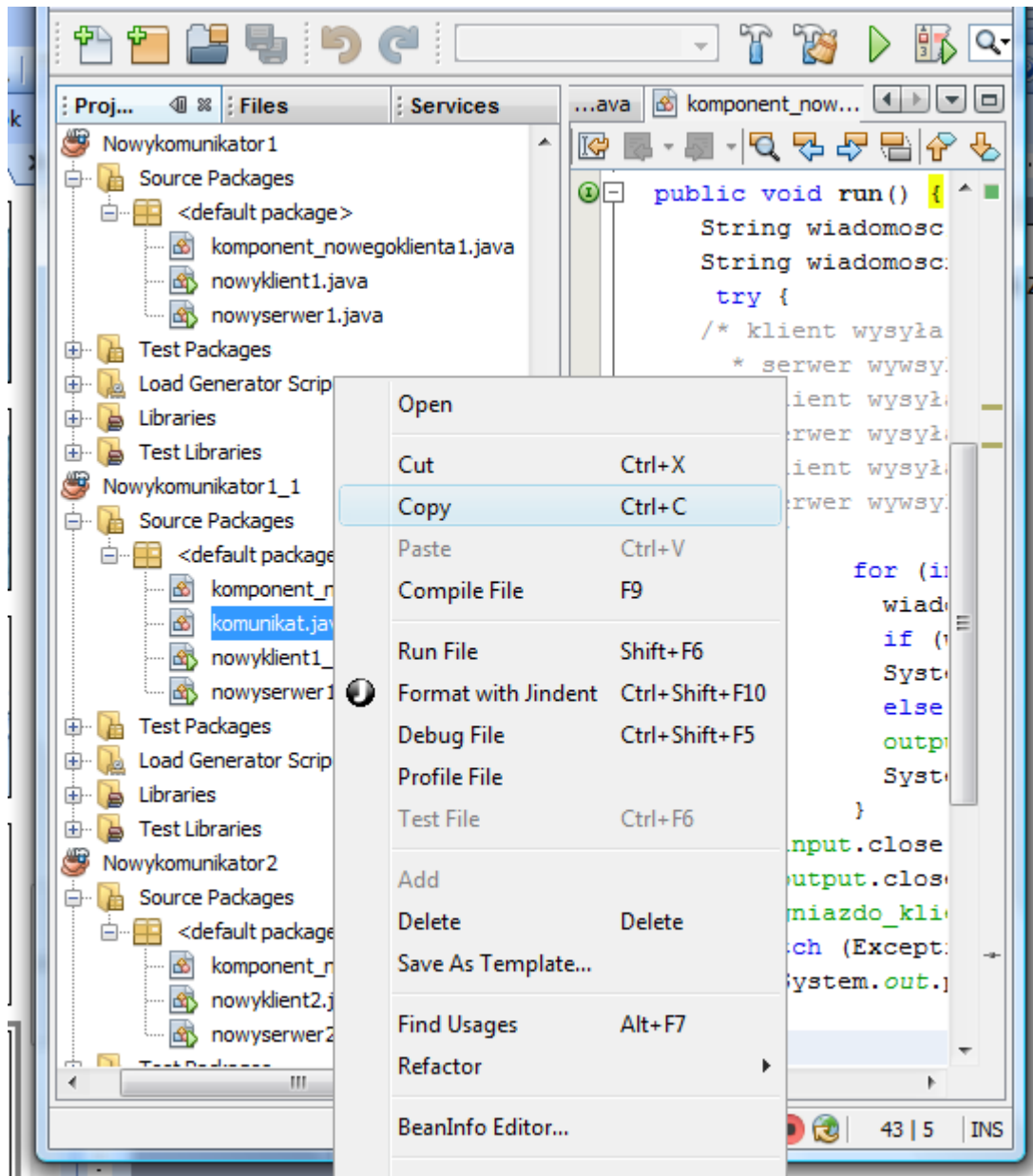
Zadanie 4 – typ komunikatu – zdefiniowany przez programistę

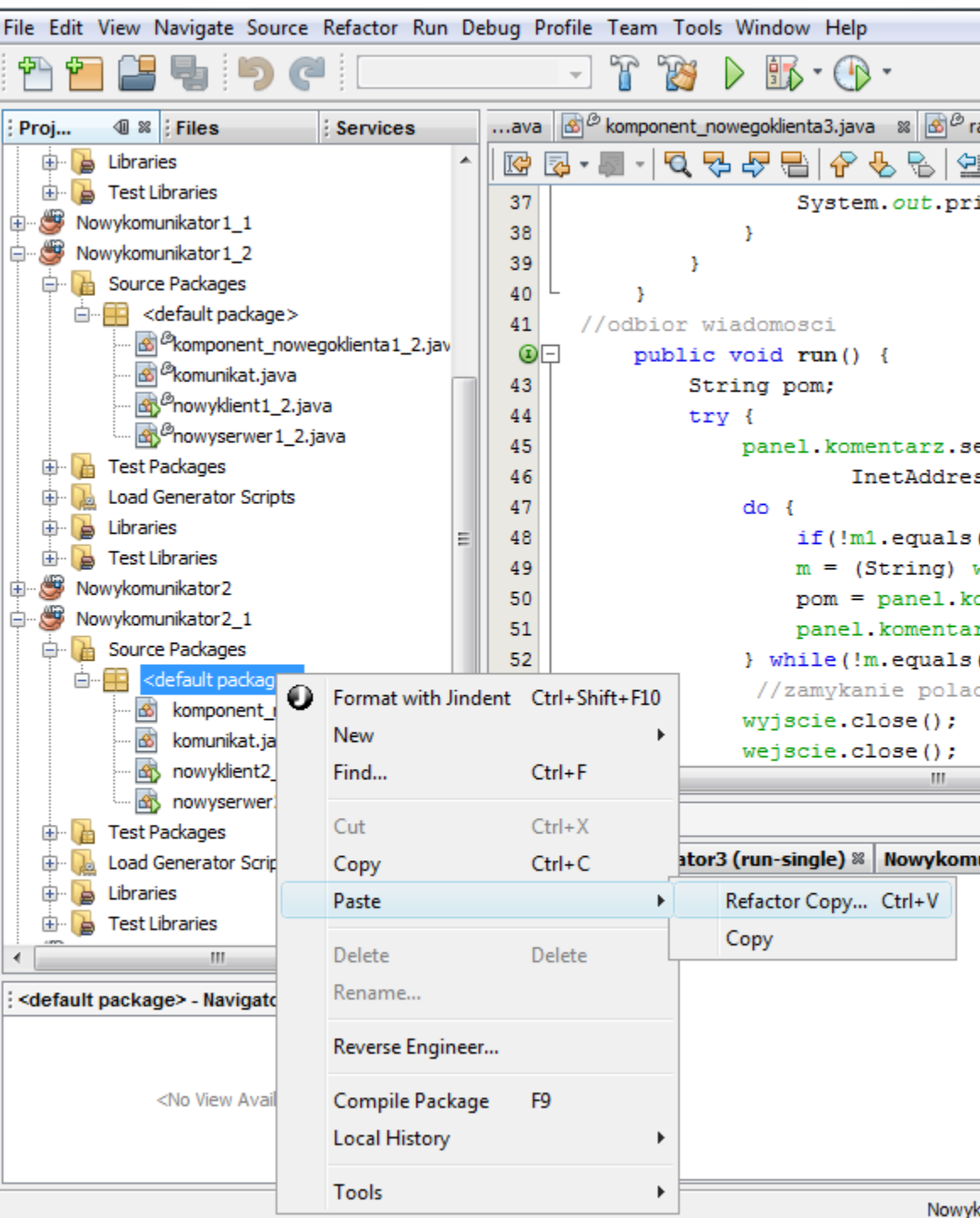
/*protokół:

- * klient wysyła A, serwer odczytuje A
- * serwer wysyła AA, klient odczytuje AA
- * klient wysyła B, serwer odczytuje B
- * serwer wysyła BB, klient odczytuje BB
- * klient wysyła C, serwer odczytuje C
- * serwer wysyła CC, klient odczytuje CC
- * */

1. Wykonaj kopię projektu **Nowykomunikator2** o nazwie **Nowykomunikator2_1**.
(slajdy 4 i 12).
2. Należy również zmienić nazwy wszystkich plików projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pliki projektu **Nowykomunikator2_1** i wybrać opcję **Refactor<Rename**. W oknie **Rename Class** zaznaczając **Apply Rename on Comments** podać odpowiednio nowe nazwy klas:
 - dla nowy klient2 podać nowy klient2_1
 - dla nowy serwer2 podać nowy serwer2_1
 - dla komponent_nowego klienta2 podać komponent_nowego klienta2_1i należy zatwierdzić nową nazwę klawiszem **Refactor**.

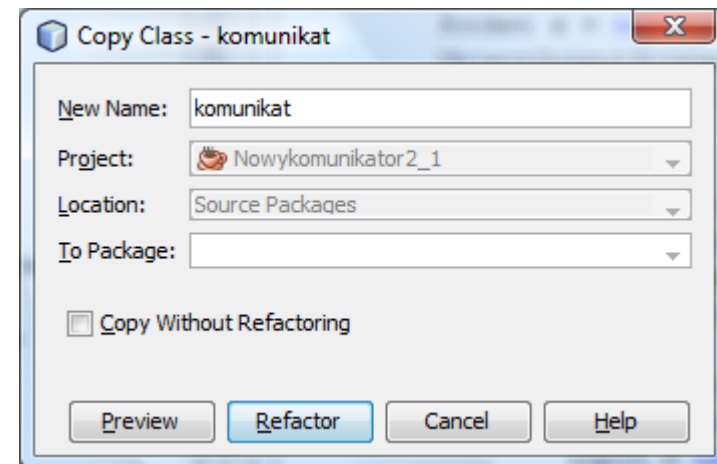
3. Należy skopiować plik **komunikat.java** z projektu **Nowykomunikat1_1**





4. Należy wkleić skopiowany w poprzednim kroku plik **komunikat.java** do projektu **Nowykomunikator2_1**. W okienku **Copy Class** należy zatwierdzić klawiszem **Refactor** kopiowanie pliku **komunikat.java**.

5. Podobnie jak w projekcie **Nowykomunikator1_1**, należy zastosować typ wiadomości **komunikat** podczas realizacji protokołu komunikacji między klientem i serwerem.



5. Uruchomienie:

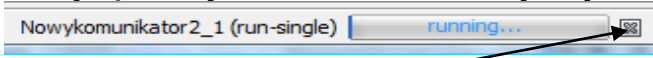
- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer2** i wybrać opcję **Run File**)

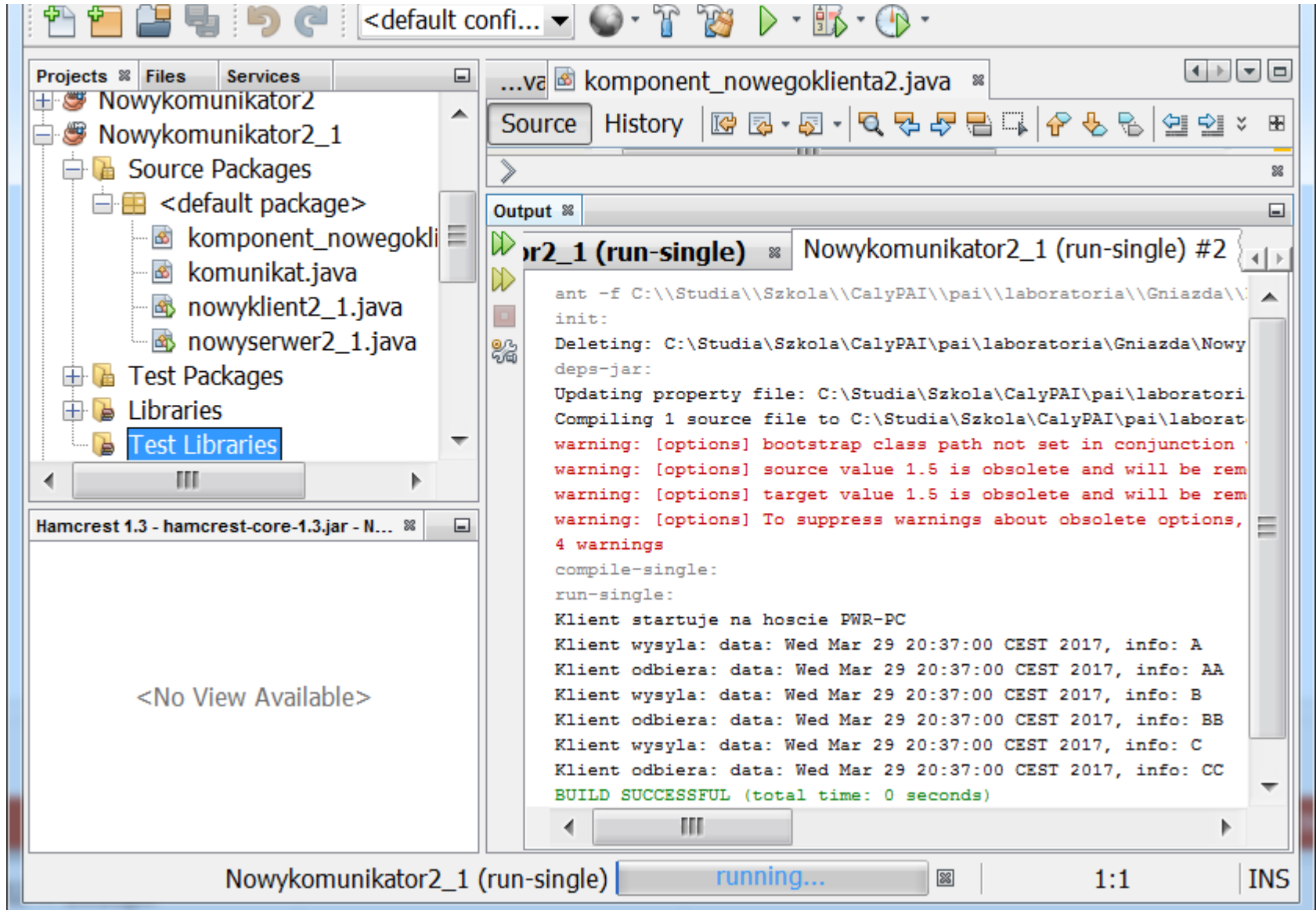
The screenshot displays an IDE interface with the following components:

- Projects View (Left):** Shows a project tree for 'Nowykomunikator2'. Under 'Source Packages', 'komponent_nowegokli...' is expanded, showing 'komunikat.java', 'nowy klient2_1.java', and 'nowyserwer2_1.java'. 'Test Libraries' is highlighted.
- Source View (Top Right):** Displays the code for 'komponent_nowegoklienta2.java', showing a 'public void run()' method.
- Output View (Bottom Right):** Shows the execution log for 'Nowykomunikator2_1 (run-single)'. The log includes directory creation, compilation warnings, and a series of server startup and data exchange messages.

```
Created dir: C:\Studia\Szkola\CalyPAI\pai\laboratoria\Gniazda\N
Created dir: C:\Studia\Szkola\CalyPAI\pai\laboratoria\Gniazda\N
Compiling 1 source file to C:\Studia\Szkola\CalyPAI\pai\laborat
warning: [options] bootstrap class path not set in conjunction
warning: [options] source value 1.5 is obsolete and will be rem
warning: [options] target value 1.5 is obsolete and will be rem
warning: [options] To suppress warnings about obsolete options,
4 warnings
compile-single:
run-single:
Serwer startuje na hoscie PWR-PC
Serwer odbiera: data: Wed Mar 29 20:36:56 CEST 2017, info: A
Serwer wysyla: data: Wed Mar 29 20:36:56 CEST 2017, info: AA
Serwer odbiera: data: Wed Mar 29 20:36:56 CEST 2017, info: B
Serwer wysyla: data: Wed Mar 29 20:36:56 CEST 2017, info: BB
Serwer odbiera: data: Wed Mar 29 20:36:56 CEST 2017, info: C
Serwer wysyla: data: Wed Mar 29 20:36:56 CEST 2017, info: CC
Serwer odbiera: data: Wed Mar 29 20:37:00 CEST 2017, info: A
Serwer wysyla: data: Wed Mar 29 20:37:00 CEST 2017, info: AA
Serwer odbiera: data: Wed Mar 29 20:37:00 CEST 2017, info: B
Serwer wysyla: data: Wed Mar 29 20:37:00 CEST 2017, info: BB
Serwer odbiera: data: Wed Mar 29 20:37:00 CEST 2017, info: C
Serwer wysyla: data: Wed Mar 29 20:37:00 CEST 2017, info: CC
```

Nowykomunikator2_1 (run-single) | running... | 1:1 | INS

- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowy klient2** i wybrać opcję **Run File**)
-  - Należy zakończyć wątek serwera, klikając na wskazany znak.



6. Zmień protokół komunikacji na inny np..

/*protokół:

- * klient wysyła A, serwer odczytuje A**
- * klient wysyła B, serwer odczytuje B**
- * serwer wysyła BB, klient odczytuje BB**
- * serwer wysyła C, klient odczytuje C**
- * klient wysyła CC, serwer odczytuje CC**
- * */**

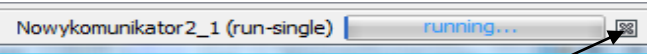
i ponownie uruchom program.

Zadanie 5

1. Uruchom program Nowykomunikator3 ()

- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer2** i wybrać opcję **Run File**)

- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowy klient2** i wybrać opcję **Run File**)

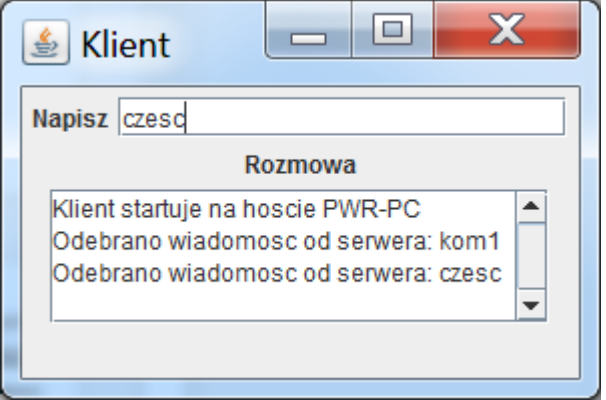
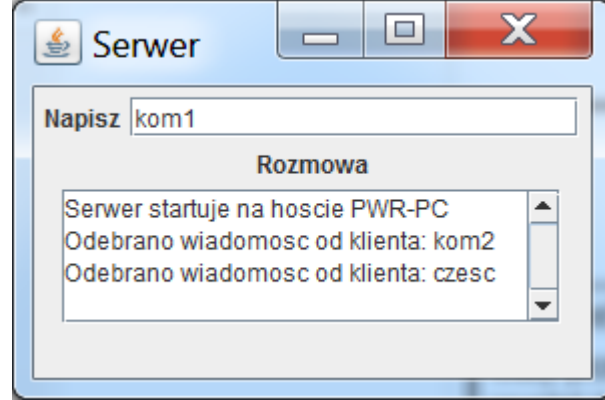
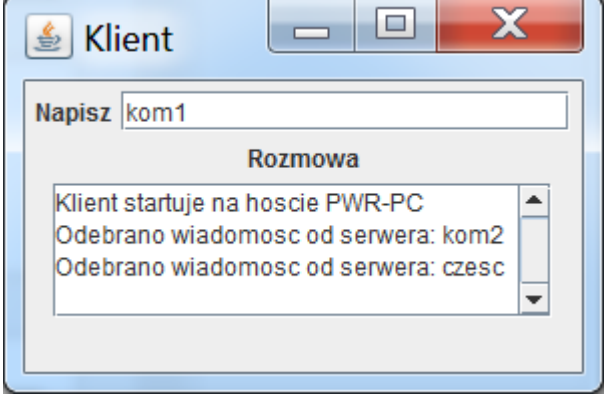
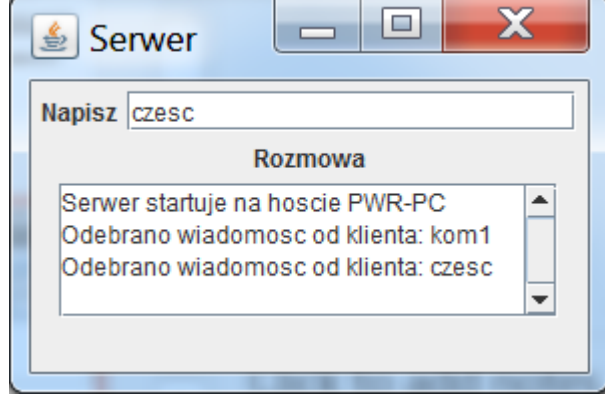
-  wskazany znak.

- Należy zakończyć wątek serwera, klikając na

2. Dokonaj analizy programu z dowolnego wykonania (podobnie jak na następnym slajdzie):

- protokołu komunikacji
- moment zakończenia protokołu

Przykład komunikacji dwóch klientów z serwerem

	klienci	Strona serwera dedykowana wielu klientom – każdy klient ma własne połączenie jako komponent klienta po stronie serwera
<p>1. Klient1</p> <p>Cały przebieg komunikacji: serwer wysłał <i>kom1</i>, klient wysłał <i>kom2</i>, klient wysłał <i>czesc</i>, serwer wysłał automatycznie <i>czesc</i> i obie strony zakończyły połączenie</p>		
<p>2. Klient2</p> <p>Cały przebieg komunikacji: klient wysłał <i>kom1</i>, serwer wysłał <i>kom2</i>, serwer wysłał <i>czesc</i>, klient wysłał automatycznie <i>czesc</i> i obie strony zakończyły połączenie</p>		

Zadanie 6 – dodatkowe

3. Dodaj do programu możliwość przechowywania komunikatów klientów w kolekcji **komunikaty**, należącej do głównego serwera **nowyserwer3**. Aby stała się wspólnym zasobem wszystkich komponentów klientów po stronie serwera (obiektów klasy **Komponent_komunikatora**), należy przekazać ją np. przez parametry nagłówka konstruktora klasy **Komponent_komunikatora** – **podczas tworzenia nowego wątku klienta po stronie serwera**. Wymaga to utworzenia konstruktora przeciążonego w tej klasie – aby zachować dotychczasowy konstruktor do tworzenia tego obiektu w celu tworzenia aplikacji komunikatora po stronie klienta, w obiekcie typu **nowy klient3**. W metodzie **run** tej klasy (**Komponent_komunikatora**) można zapisywać te komunikaty do tej wspólnej kolekcji np. w bloku synchronizowanym (metody kolekcji ArrayList nie są synchronizowane) po odbiorze kolejnego komunikatu od klienta.
4. Polecenia wyświetlania tych komunikatów można wykonać dodając np. MenuBar do okienka JFrame w klasie ramka, i menu rozwijane do elementów MenuBar. Jedna z pozycji menu może zostać wykorzystana do wysłania polecenia przez klienta do serwera (do obiektu typu **Komponent_komunikatora**) o wysłaniu zawartości kolekcji **komunikaty**. **Po stronie serwera w metodzie run należy rozpoznać to polecenie po odczytaniu wiadomości i wtedy** zamiast zapisywać kolejny komunikat do kolekcji **komunikaty**, wysłać zawartość kopii tej kolekcji do klienta. Po stronie klienta w metodzie **run** należy rozpoznać typ nadesłanej informacji i wyświetlić zawartość nadesłanej kolekcji **komunikaty** na ekranie klienta, jeśli został rozpoznany taki typ odebranej informacji.