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**:

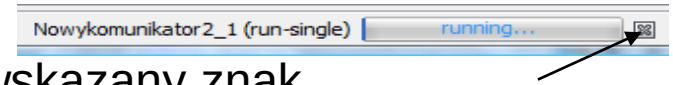
```
gniazdo_klienta = serwer.accept();
```

oraz

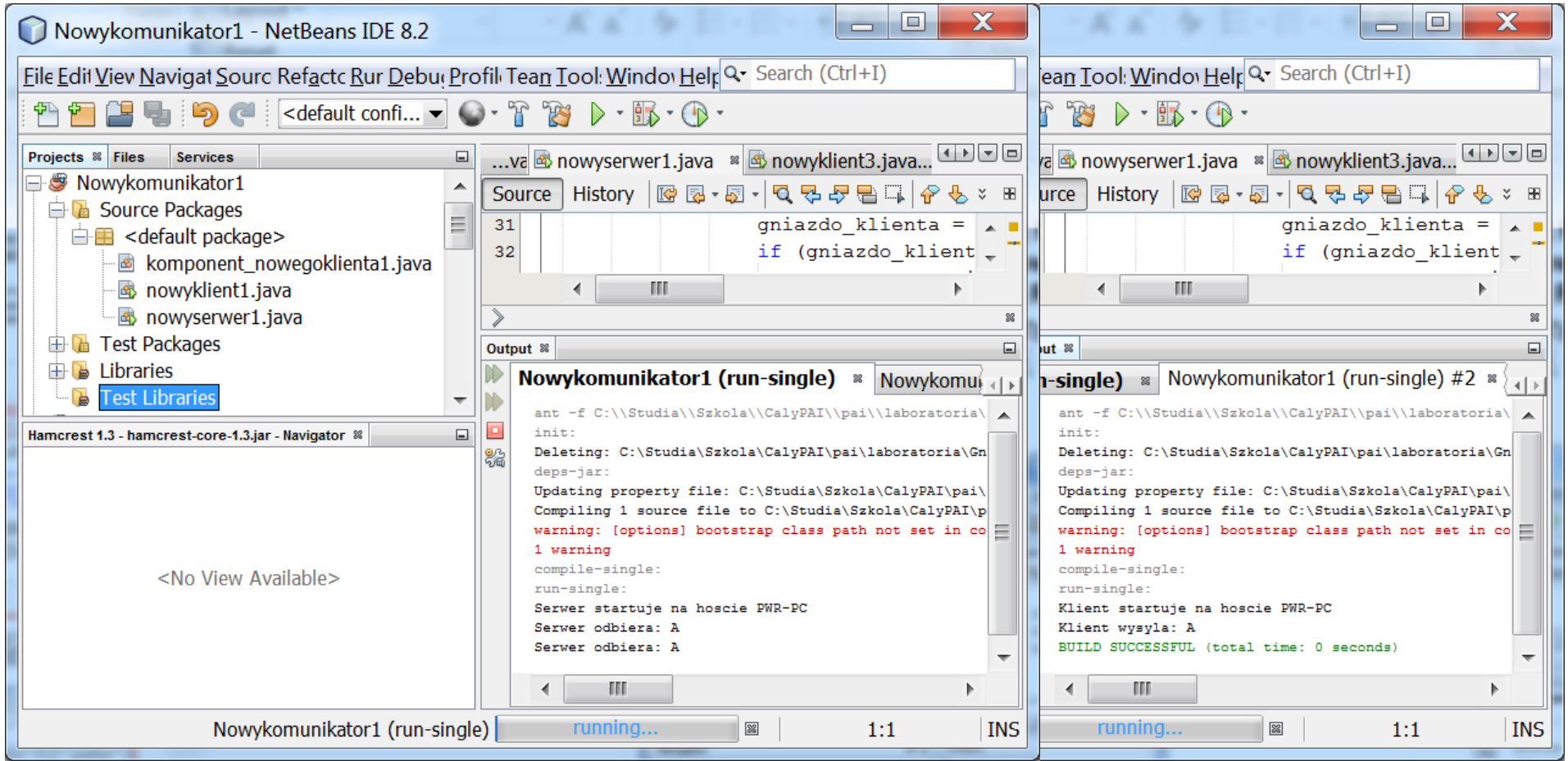
```
wyjscie = new ObjectOutputStream(gniazdo_klienta.getOutputStream());
wyjscie.flush();
wejscie = new ObjectInputStream(gniazdo_klienta.getInputStream());
komponent_nowegoklienta1 komp_klienta =
        new komponent_nowegoklienta1(gniazdo_klienta, wejscie, wyjscie);
Thread watek_komponentu_klienta = new Thread(komp_klienta);
watek_komponentu_klienta.start();
```

3. Wyjaśnij rolę klasy **nowyklient1** 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:

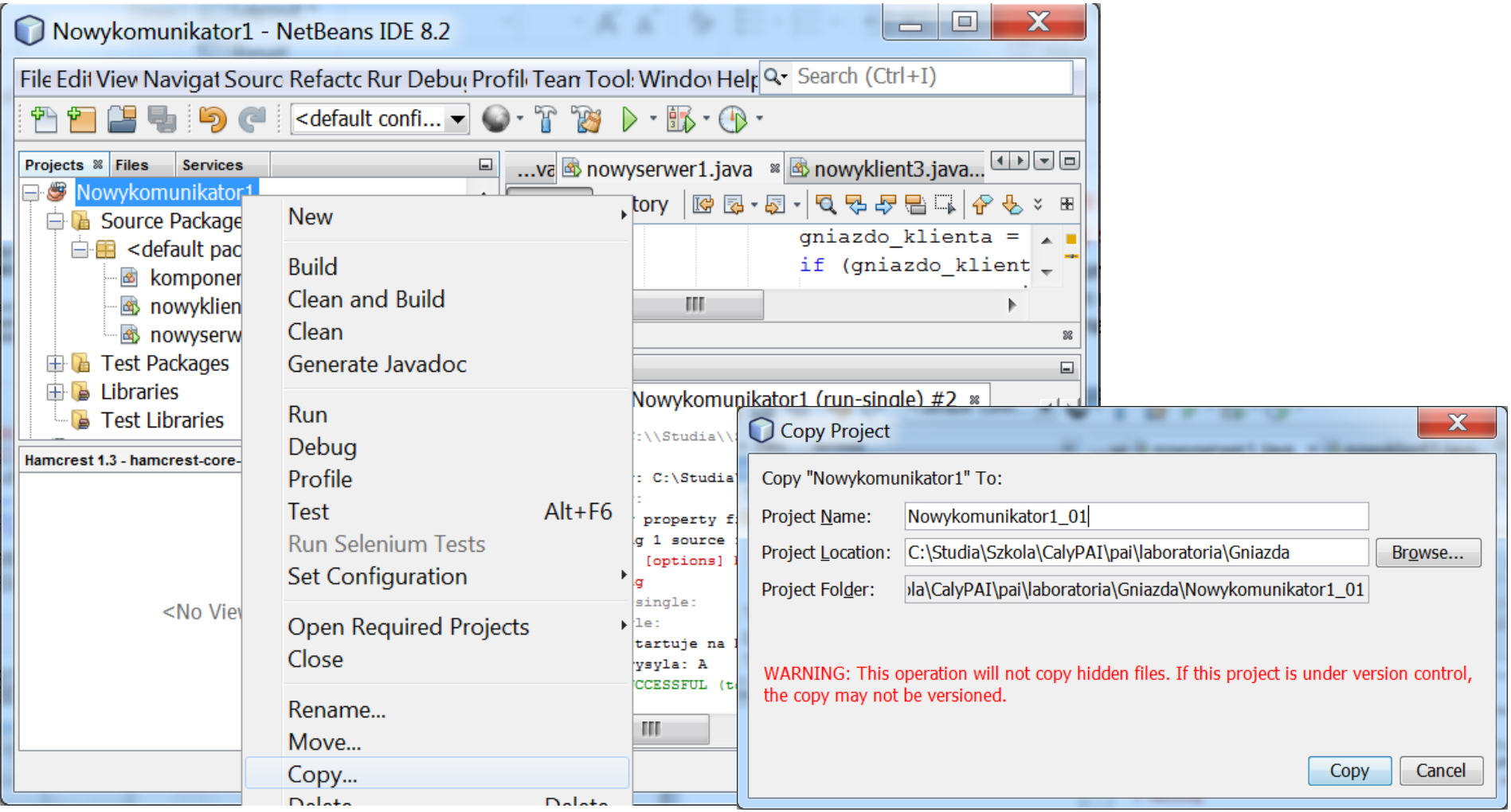
- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer1** i wybrać opcję **Run File**)
- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy:w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyklient1** i wybrać opcję **Run File**)
- 

Aby zakończyc program serwera, należy kliknąć na wskazany znak.



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

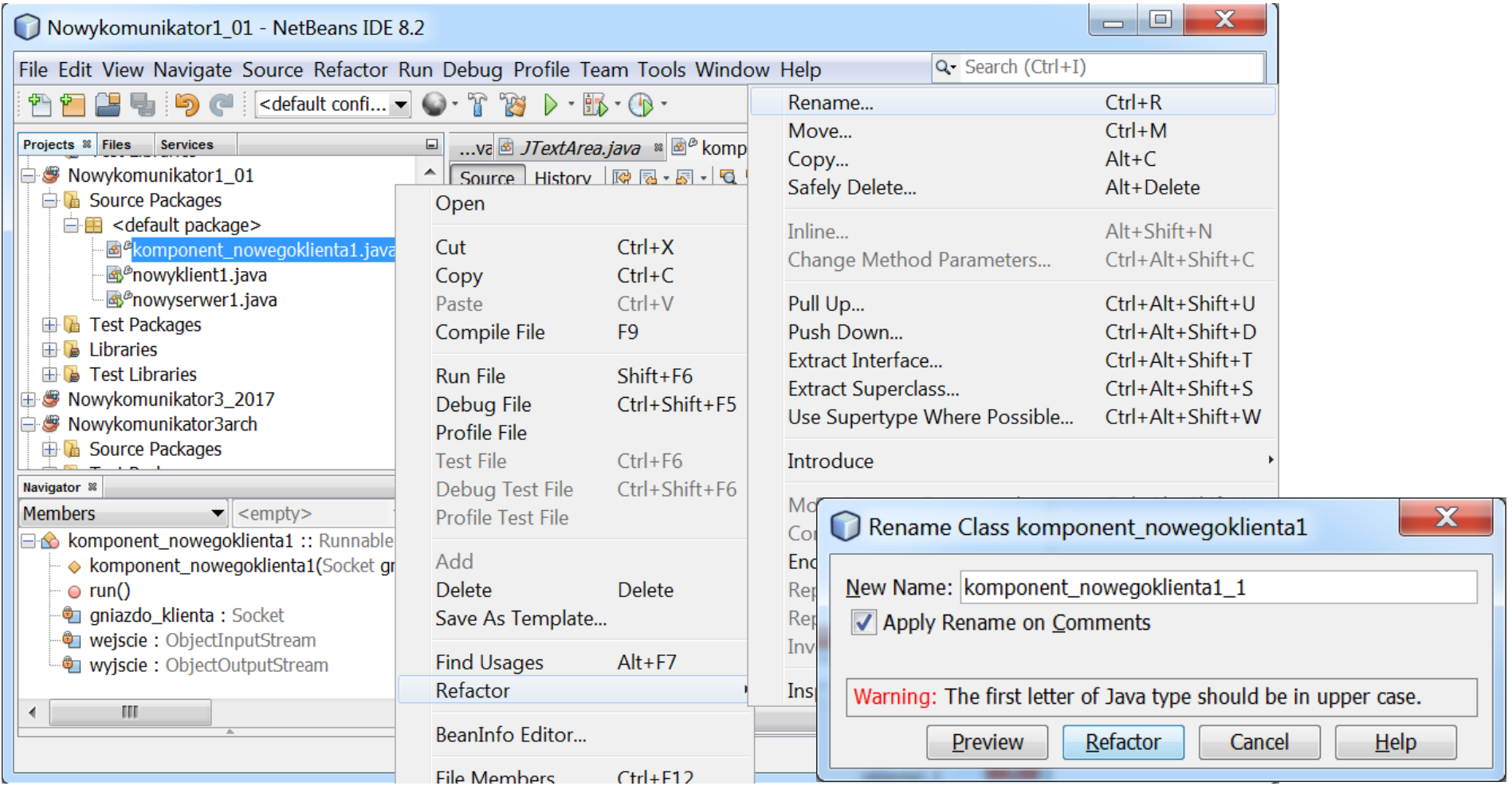
1. Wykonaj kopię programu Nowykomunikator1 jako Nowykomunikator1_01 - w oknie **Projects** należy kliknąć prawym klawiszem myszy na projekt **Nowykomunikator1** i wybrać opcję **Copy**. W okienku **Copy Projekt** podać nazwę nowego projektu (Project Name) oraz położenie (Project Location).



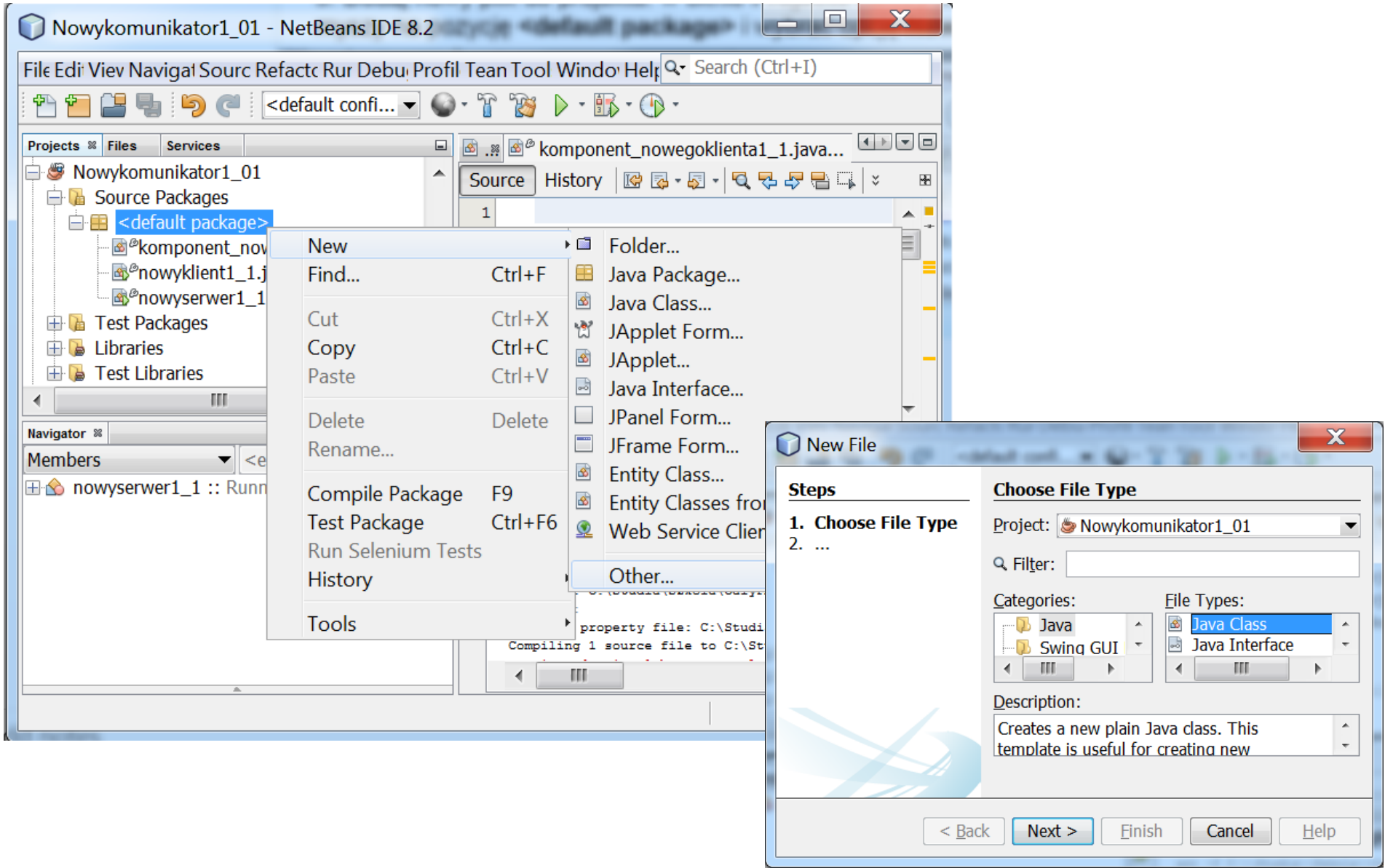
2. Należy również zmienić nazwy wszystkich plików projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pliki projektu **Nowykomunikator1_1** i wybrać opcję **Refactor<Rename**. W oknie **Rename Class** zaznaczając **Apply Rename on Comments** podac odpopwiednio nowe nawy klas:

- dla nowyklient1 podac nowyklient1_1
- dla nowyserwer1 podać nowyserwer1_1
- dla komponent_nowegoklienta1 podać komponent_nowegoklienta1_1

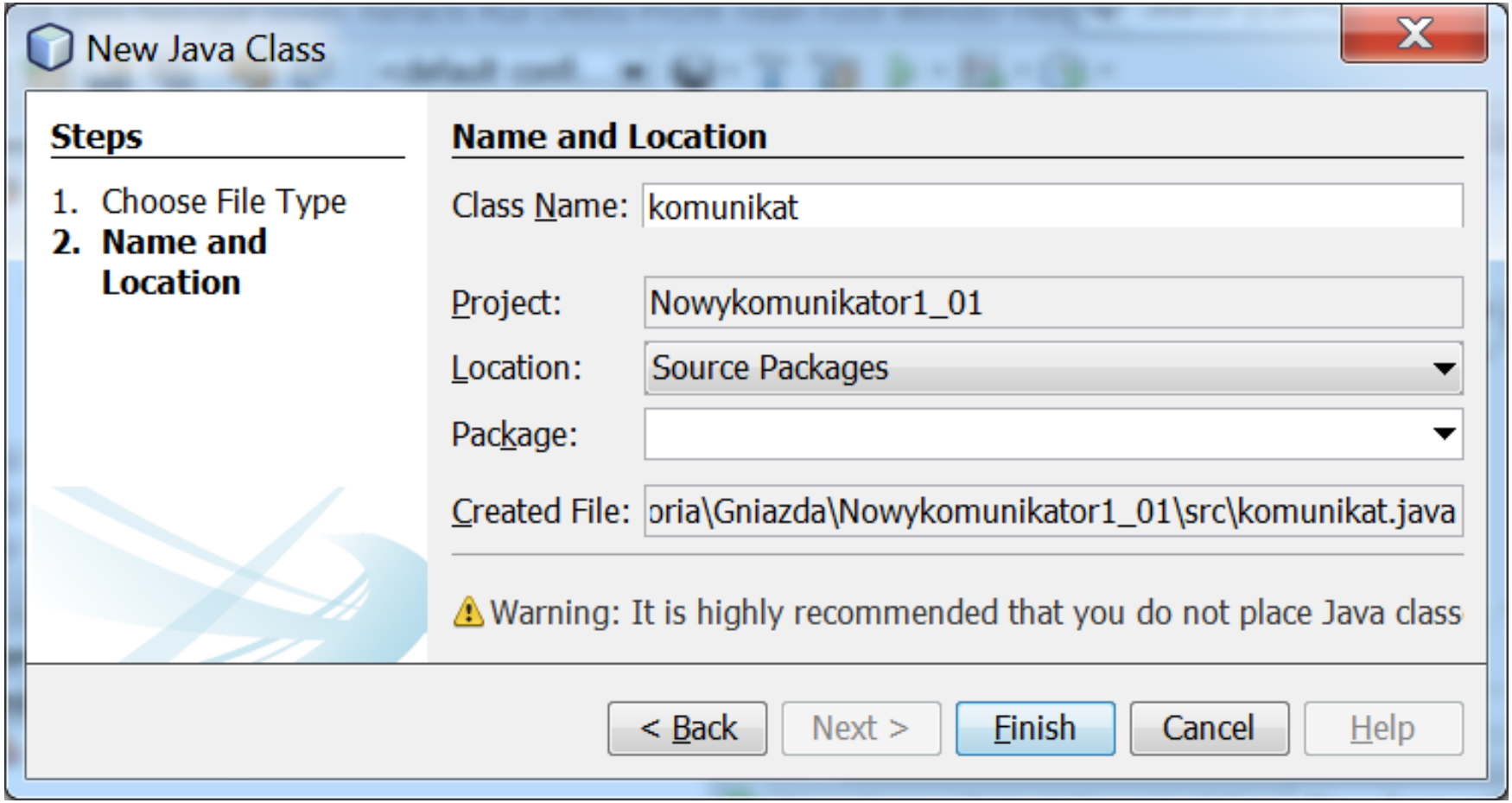
i należy zatwierdzić nową nazwę klawiszem **Refactor**.



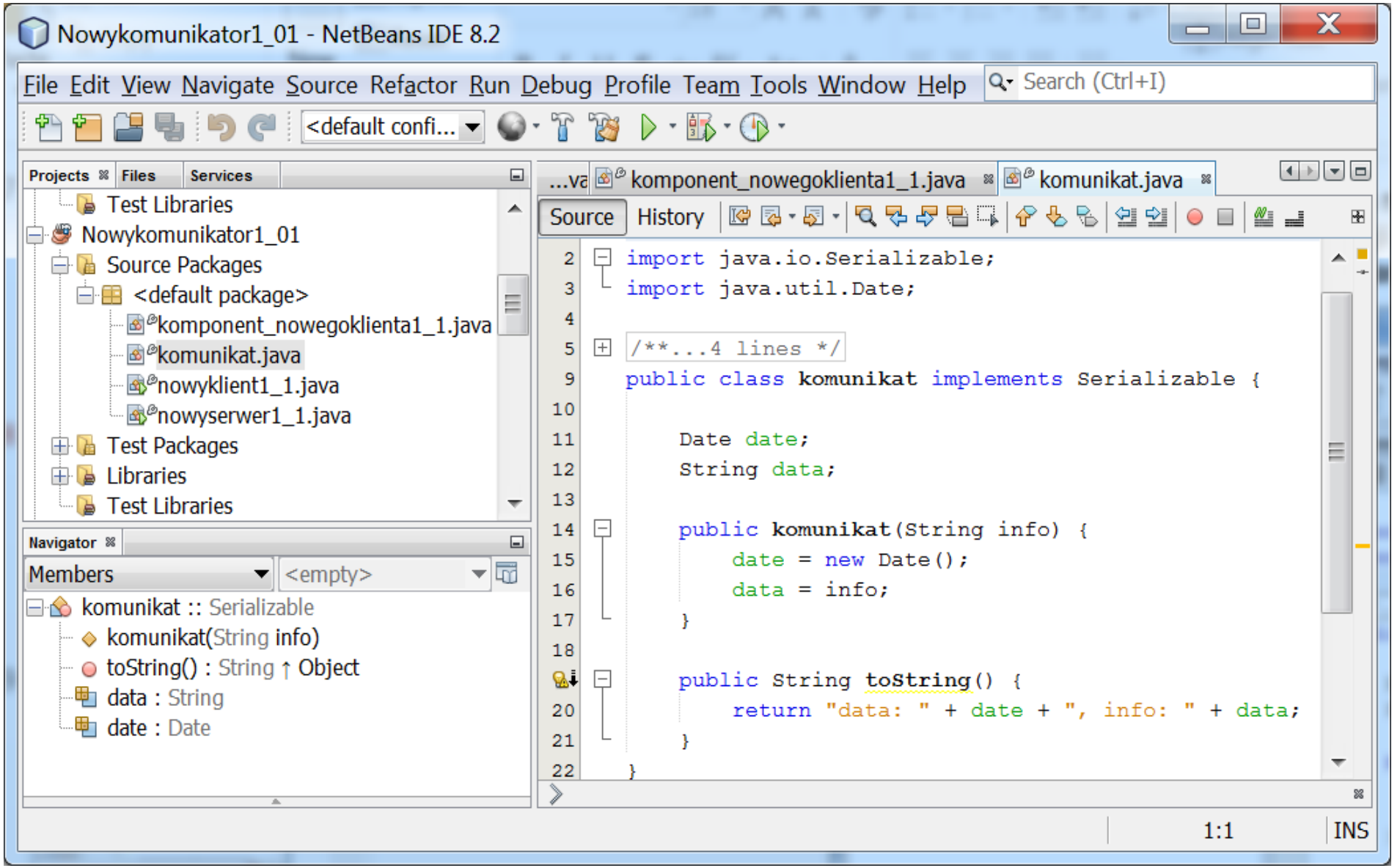
3. Dodaj nowy plik do projektu: w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **<default package>** i wybrać opcję **New/Other/Java/Java Class**



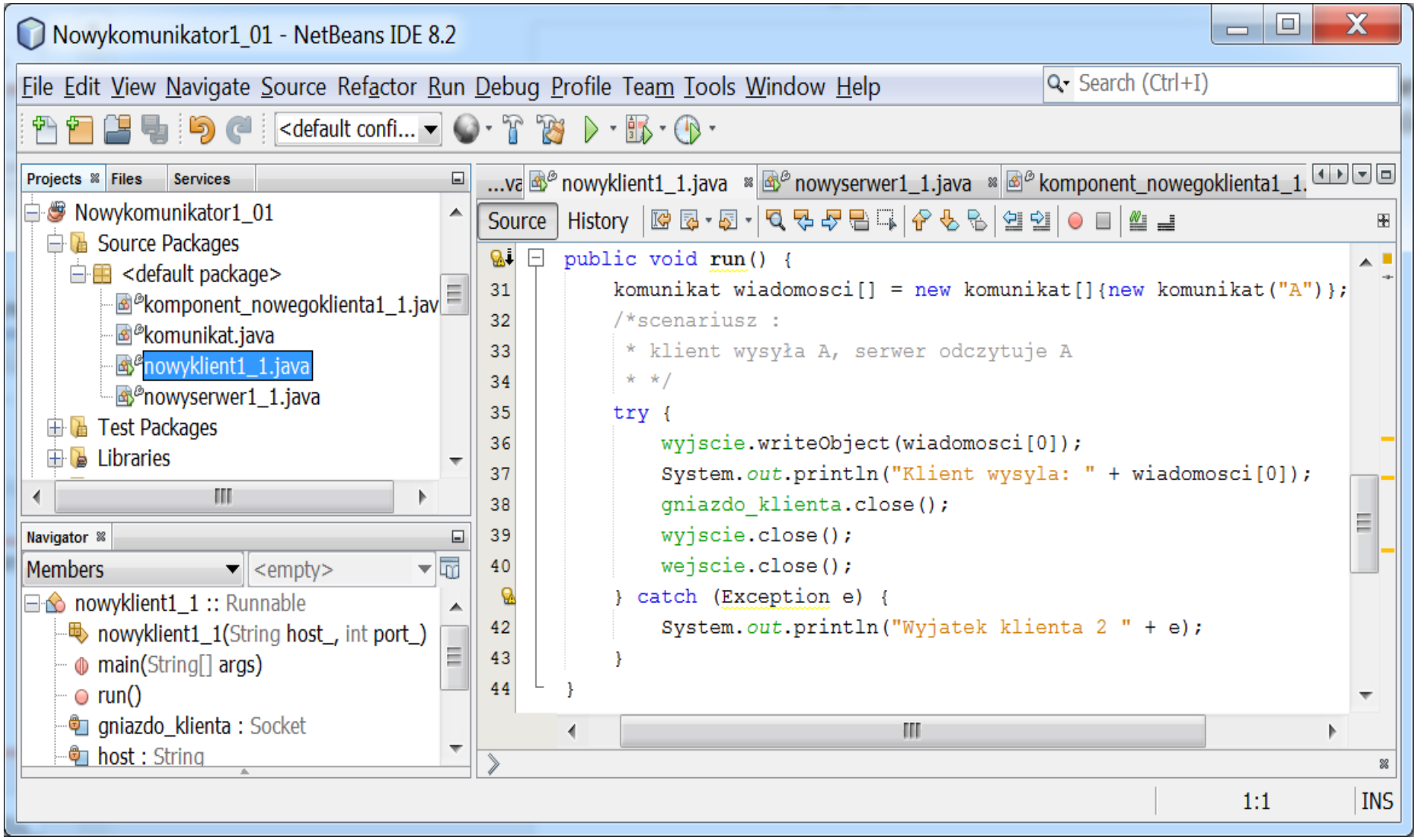
W okienku **New Java Class** nadac nowej klasie nazwę **komunikat**



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

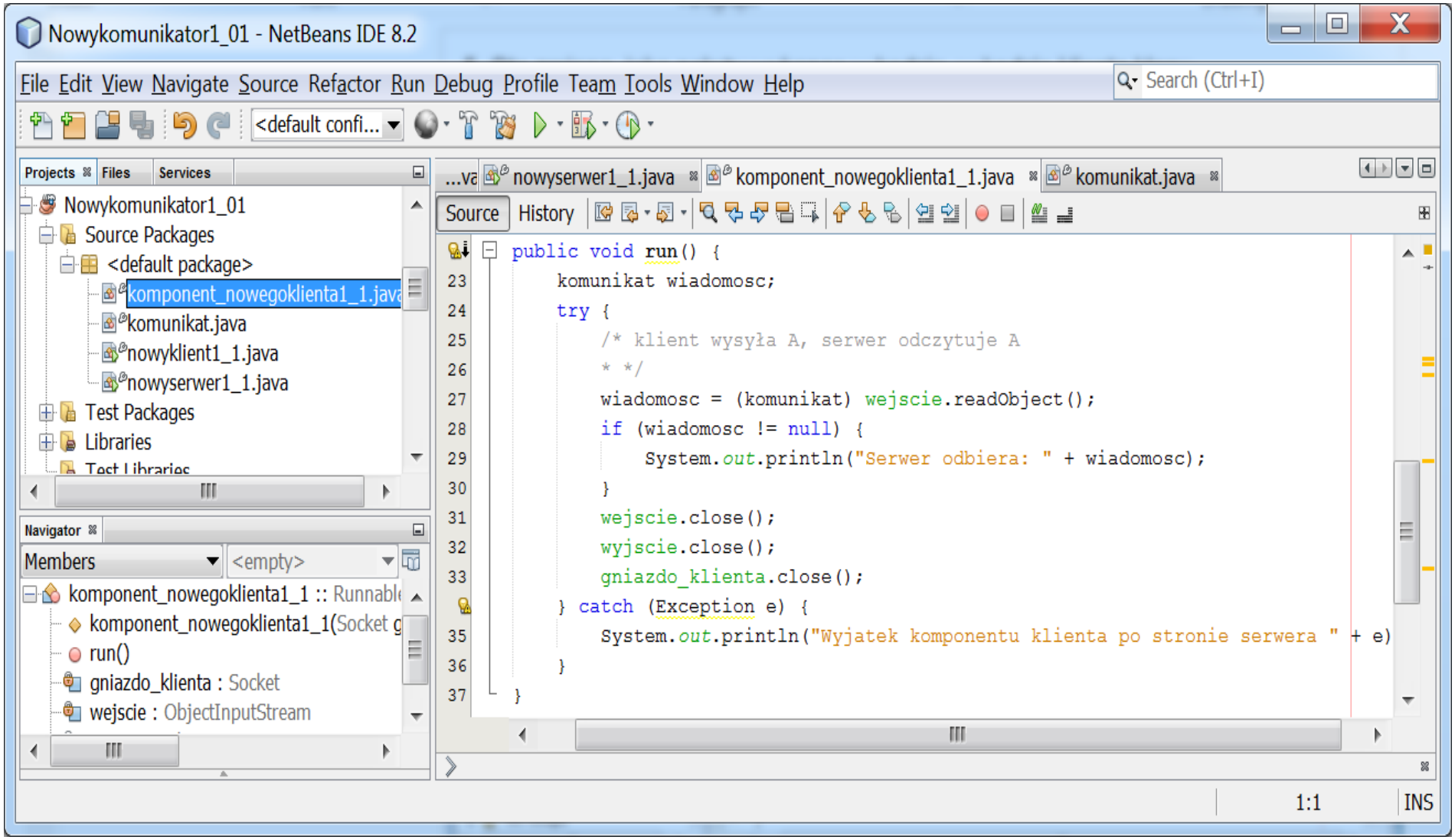


4. Oto zmiana, jaka należy wykonac w kodzie w kodzie klienta klasy **nowyklient1_1.** Należy odpowiednio zmienić kod metody **run** klasy komponent_nowegoklienta1_1, zmieniając typ odbieranego komunikatu z typu **String** na typ **komunikat**



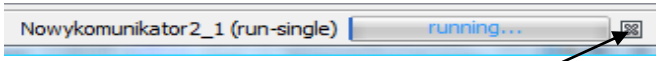
5. Oto zmiana, jaka należy wykonac w kodzie w kodzie klienta klasy **komponent_nowegoklienta1_1.**

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

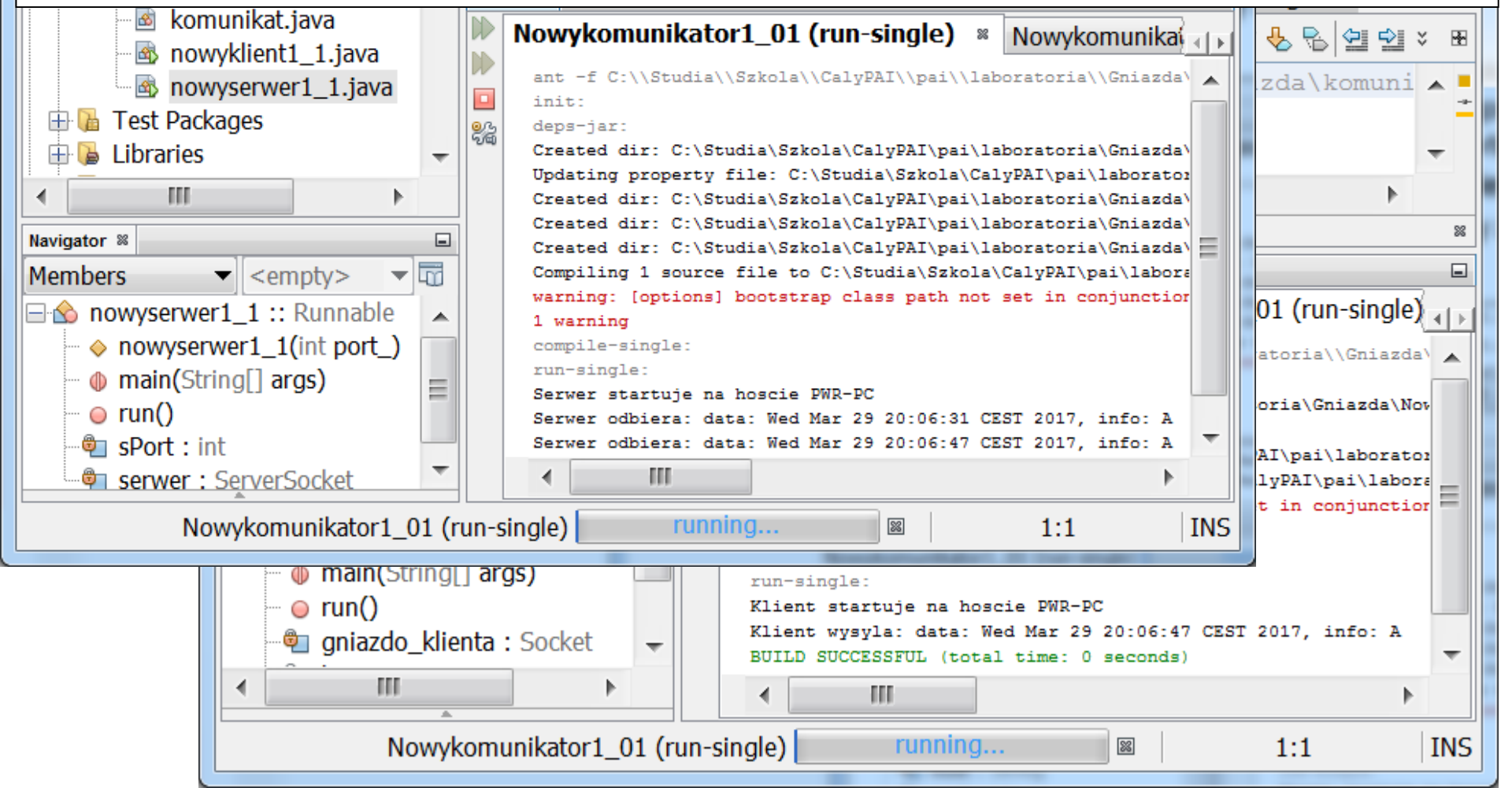


6. Uruchomienie:

- 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 **nowyklient1_1** i wybrać opcję **Run File**)



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



Zadanie 3 – typ komunikatu - String

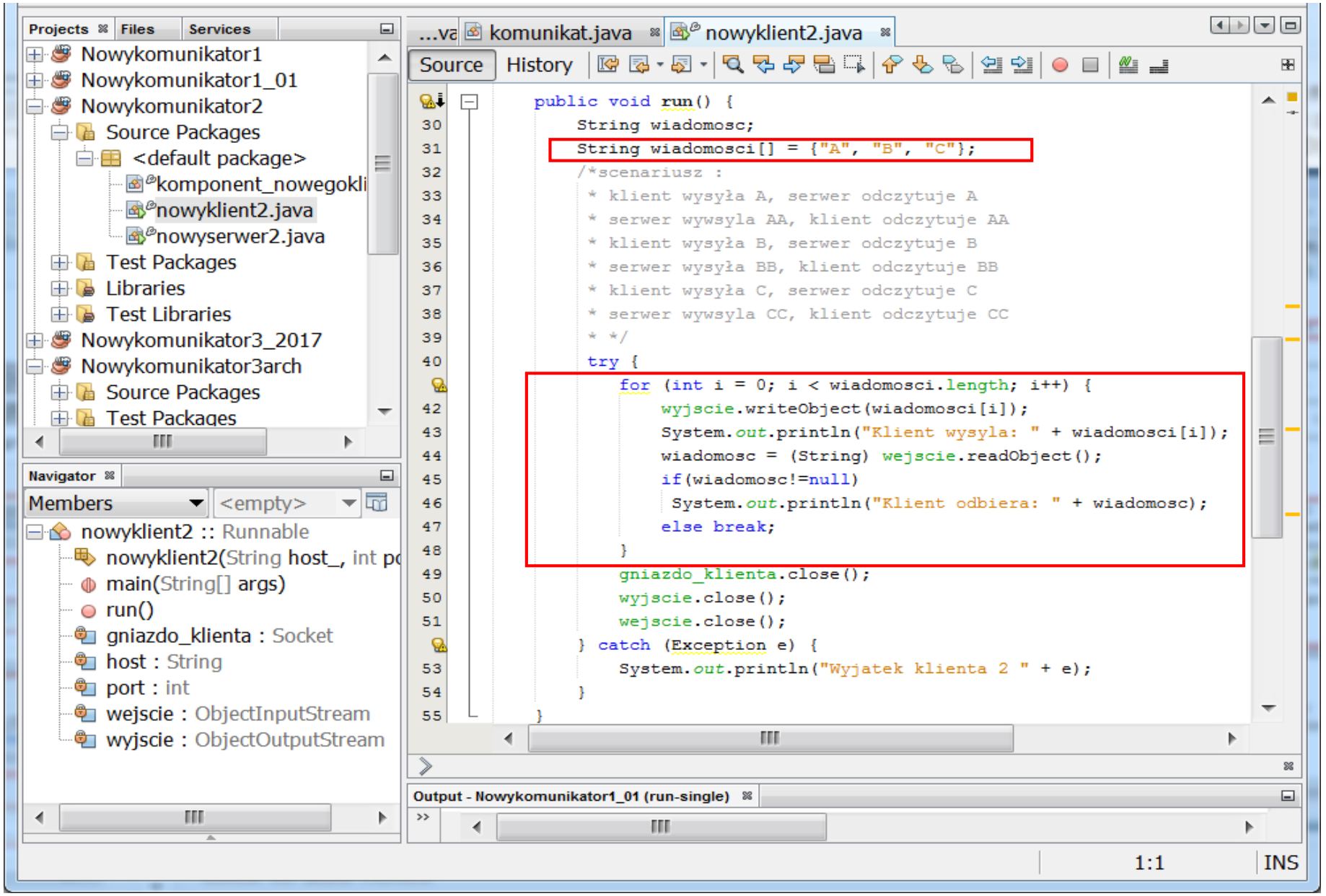
/*protokół:

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

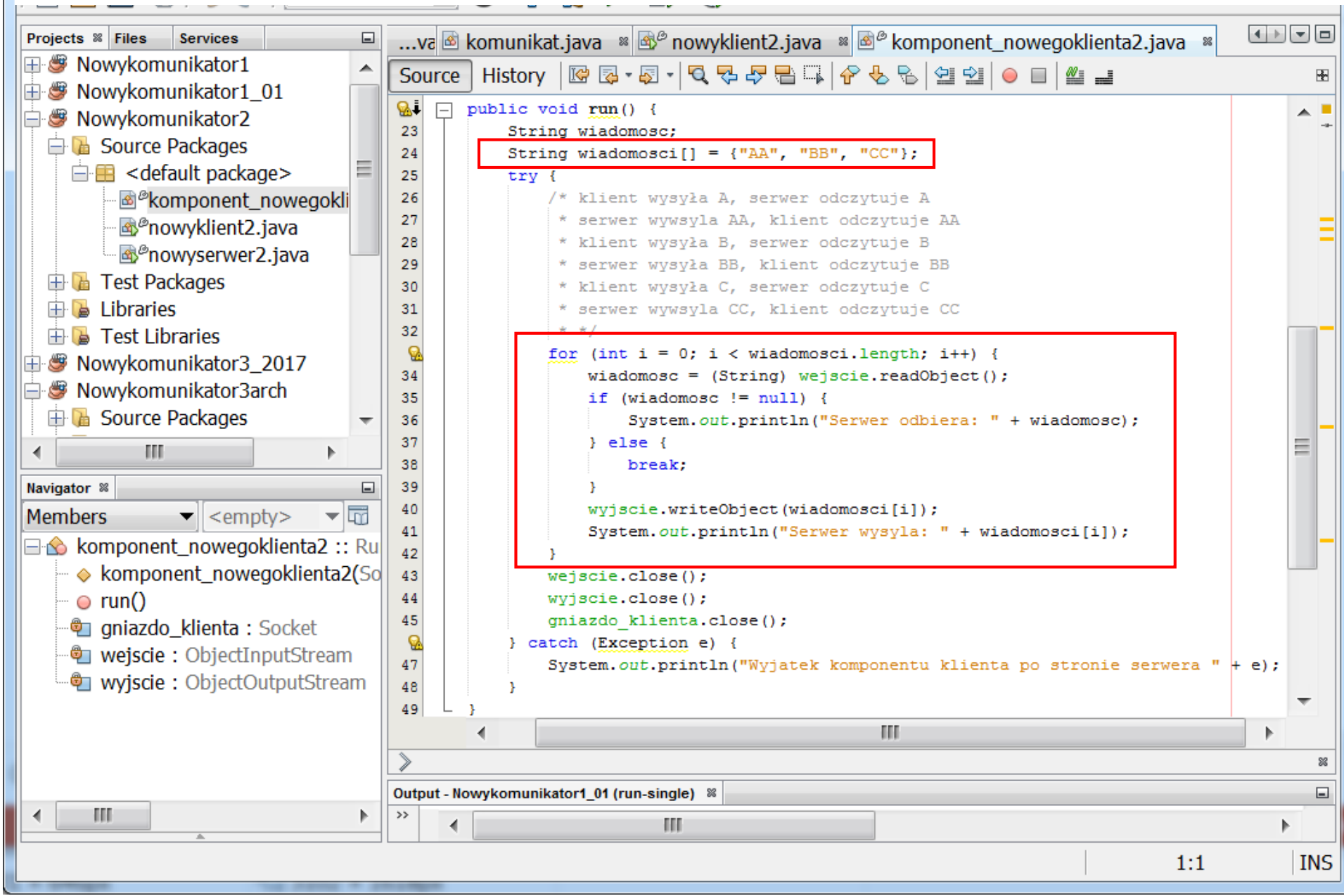
1. Wykonaj program **Nowykomunikator2,** który realizuje protokół wymiany komunkatów, podany na kolejnym slajdzie, jako kopię projektu **Nowykomunikator1** - w oknie **Projects** należy kliknąć prawym klawiszem myszy na projekt **Nowykomunikator1** i wybrać opcję **Copy**. W okienku **Copy Projekt** 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 odpopwiednio nowe nawy klas:
   - dla nowyklient1 podac nowyklient2
   - dla nowyserwer1 podać nowyserwer2
   - dla komponent_nowegoklienta1 podać komponent_nowegoklienta2

   i należy zatwierdzić nową nazwę klawiszem **Refactor**.

3. Zmiana kodu klasy klienta **nowyklient2** – w metodzie **run** dokonano zmiany protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **OutputObjectStream** oraz **InputObjectStream**

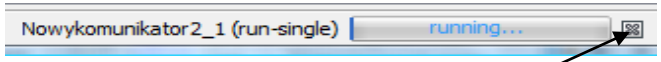


4. Zmiana kodu klasy klienta **komponent_nowegoklienta2** – w metodzie **run** dokonano zmiany protokołu wymienianych komunikatów między klientem i serwerem za pomocą metod **writeObject** i **readObject** strumieni **OutputObjectStream** oraz **InputObjectStream**

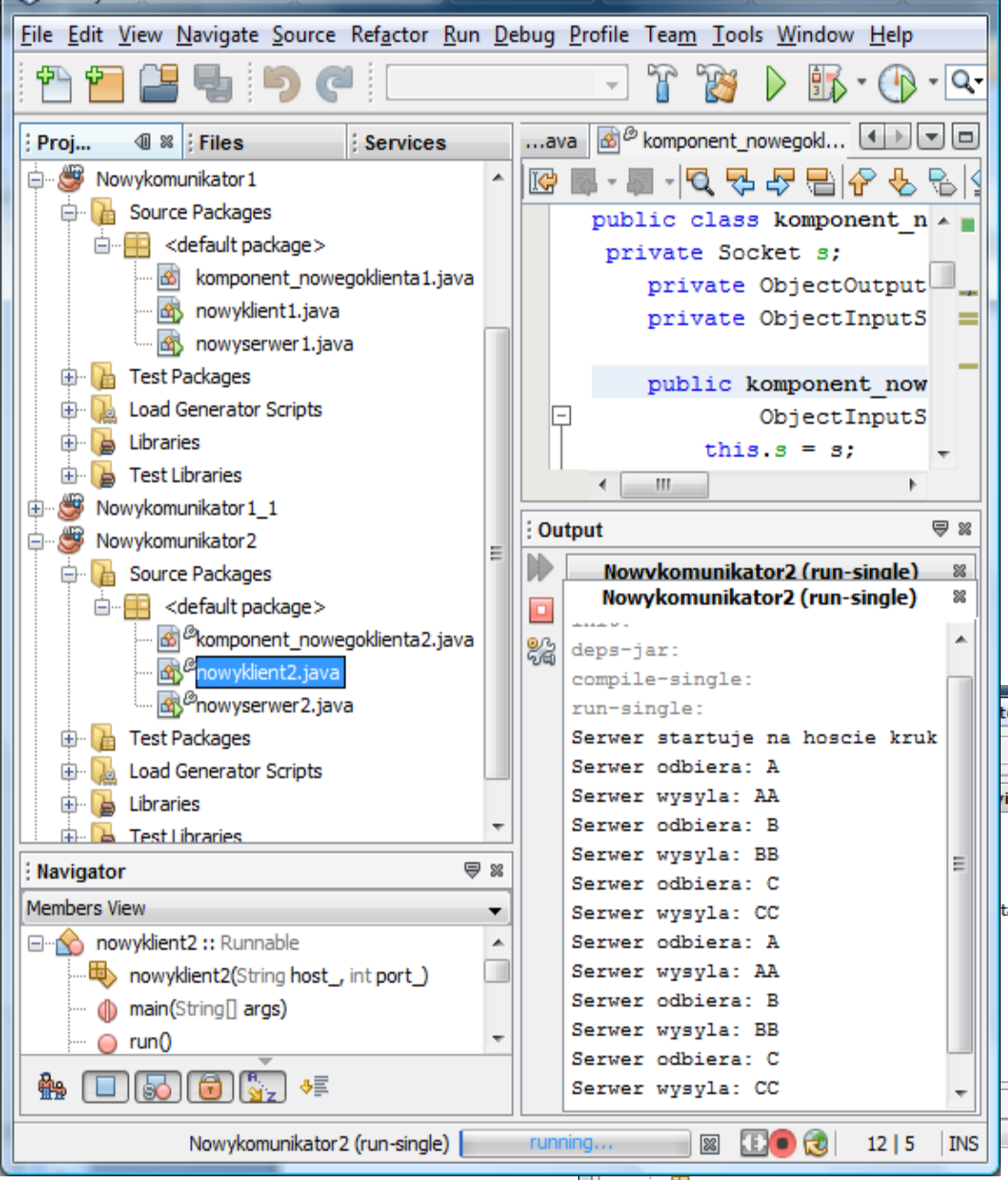


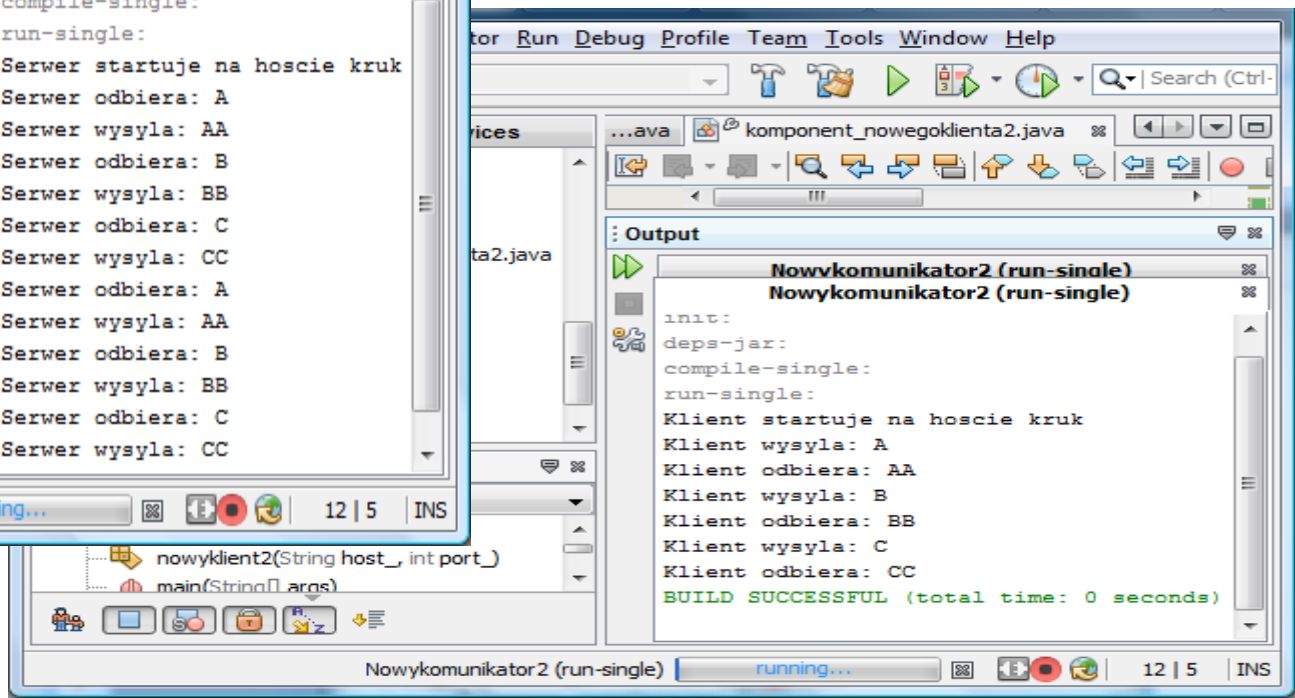
5. Uruchomienie:

- 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 **nowyklient2** 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

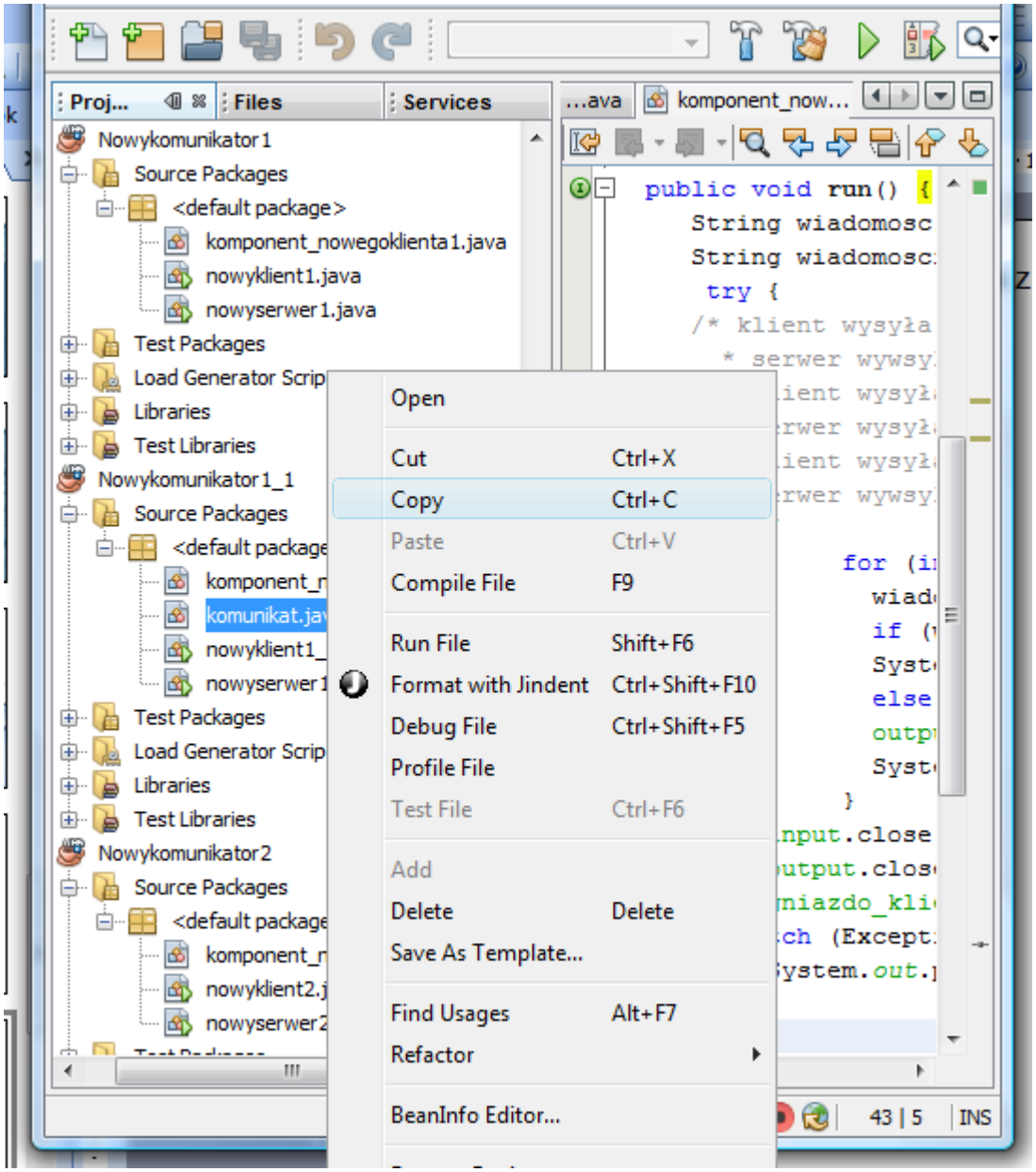
/*protokół:

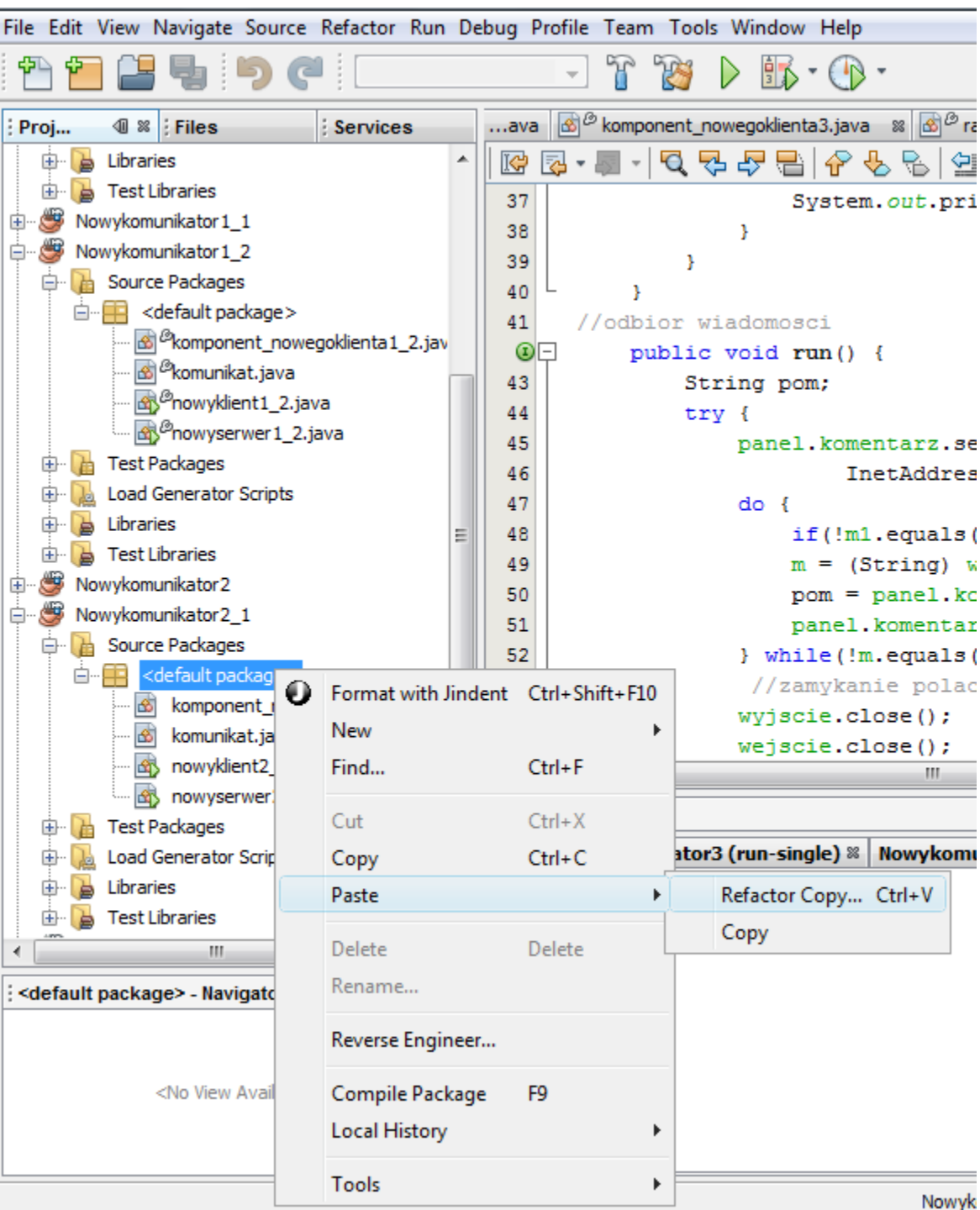
*** klient wysyła A, serwer odczytuje A**
*** serwer wysyla AA, klient odczytuje AA**
*** klient wysyła B, serwer odczytuje B**
*** serwer wysyła BB, klient odczytuje BB**
*** klient wysyła C, serwer odczytuje C**
*** serwer wysyla CC, klient odczytuje CC**
*** */**

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 odpopwiednio nowe nawy klas:
   - dla nowyklient2 podac nowyklient2_1
   - dla nowyserwer2 podać nowyserwer2_1
   - dla komponent_nowegoklienta2 podać komponent_nowegoklienta2_1

   i należy zatwierdzić nową nazwę klawiszem **Refactor**.

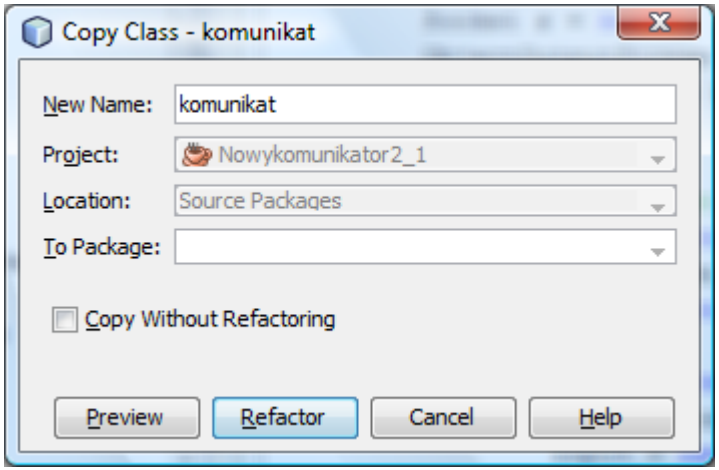
3. Należy skopiować plik **komunikat.java** z projektu **Nowykomunikat1_1**





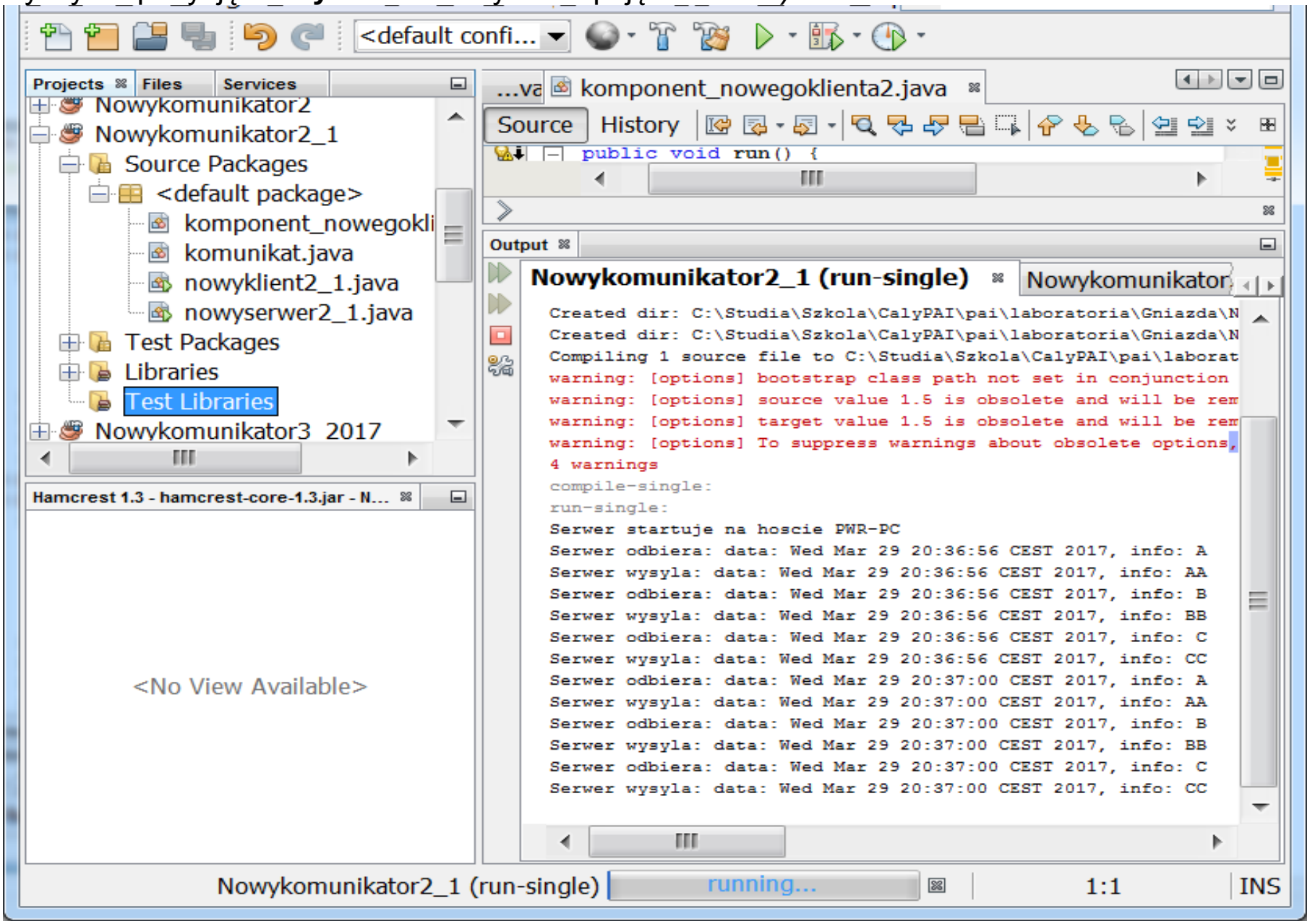
4. Należy wkleić skopiowany w poprzednim kroku plik **komunikat.java** do projektu **Nowykomunikator2_1.** W okienku **Copy Class** należy zatwierdzić klawiszem **Refactor** kopiowanie pliku **komunikat.java**.

5. Podobnie jak w projekcie Nowykiomunikator1_1, należy zastosować typ wiadomości **komunikat** podczas realizacji protokołu komunikacji między klientem i serwerem.

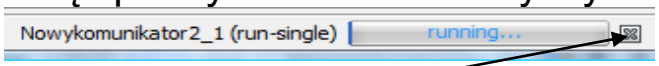


5. Uruchomienie:

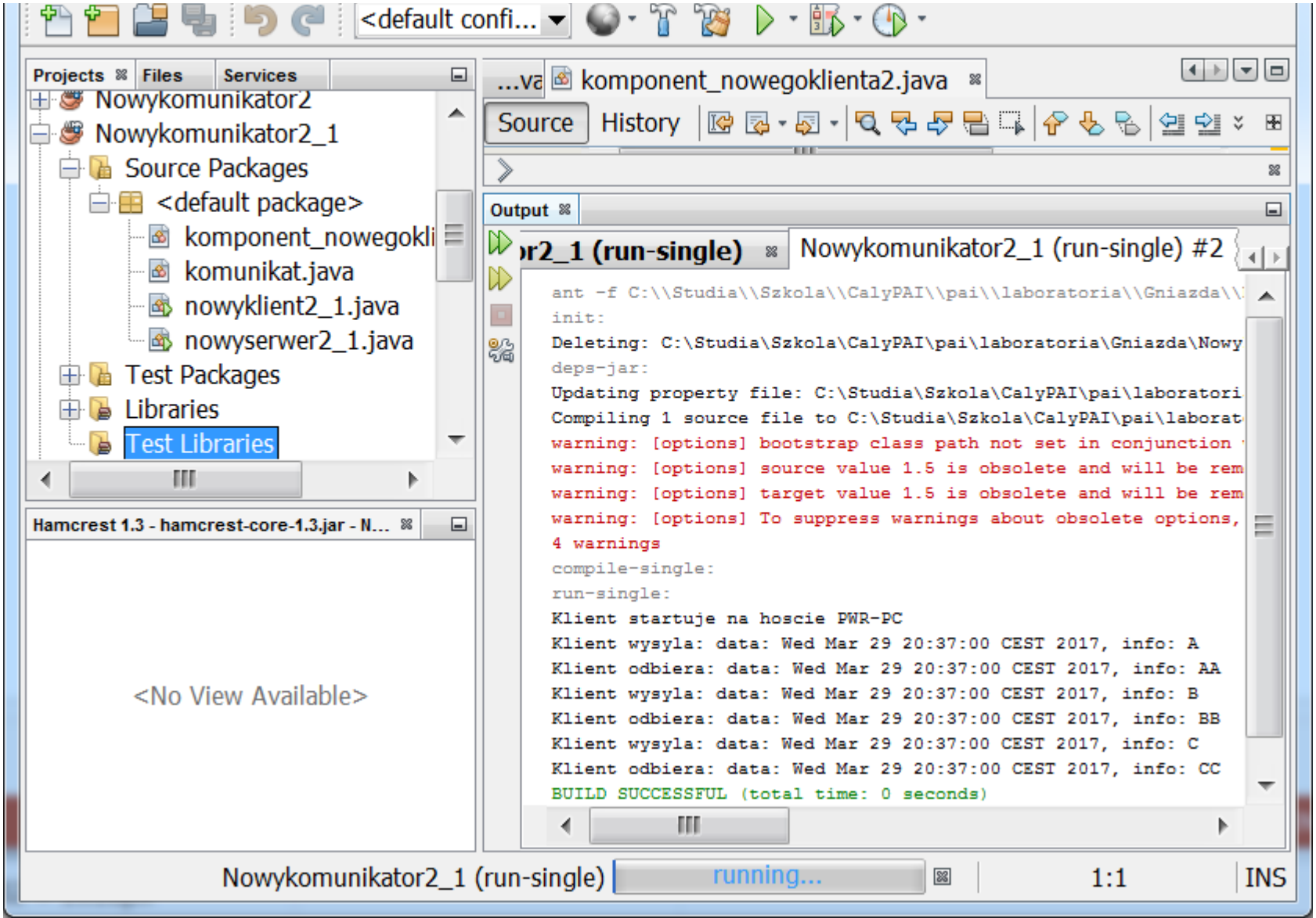
- 1 program serwera za pomocą **Run File** (w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyserwer2** i wybrać opcję **Run File**)



- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy:w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyklient2** 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 wysyla CC, serwer odczytuje CC**

*** */**

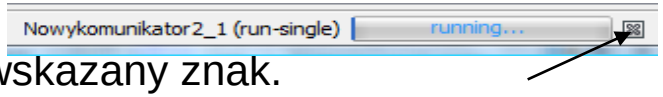
i ponownie uruchom program.

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**)

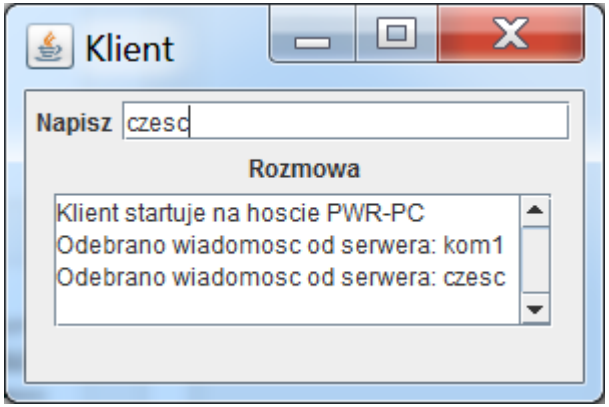
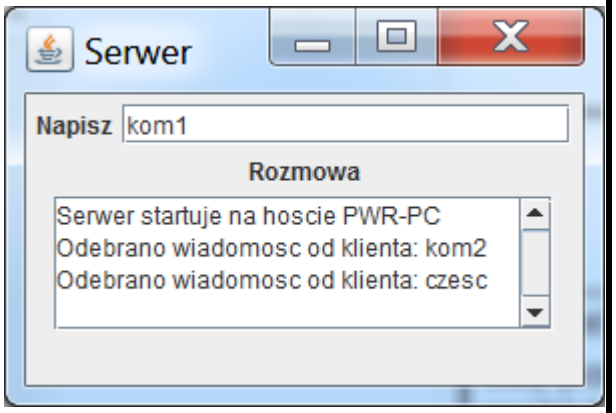
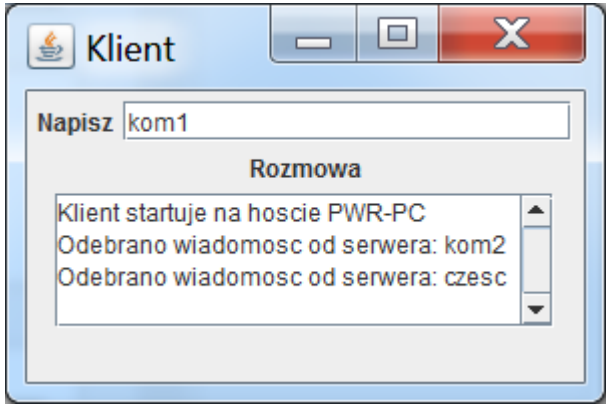
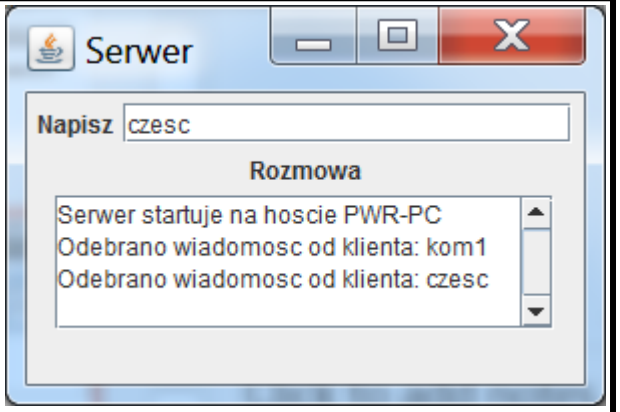
- 2 programy klienta za pomocą **Run File** (powtórzyć dwa razy:w oknie **Projects** należy kliknąć prawym klawiszem myszy na pozycję **nowyklient2** i wybrać opcję **Run File**)

- 

- Należy zakończyć wątek serwera, klikając na wskazany znak.

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**
1. Klient1 Cały przebieg komunikacji: serwer wysłał *kom1*, klient wysłał *kom2*, klient wysłał *czesc*, serwer wysłał automatycznie *czesc* i obie strony zakończyły połączenie		
2. Klient2 Cały przebieg komunikacji: klient wysłał *kom1*, serwer wysłał *kom2*, serwer wysłał *czesc*, klient wysłał automatycznie *czesc* i obie strony zakończyły połączenie		

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 **nowyklient3**. 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 zapisywac 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.