

Kontynuacja programu z lab2 – refaktoryzacja kodu oparta na dodaniu obiektu transferowego oraz stronicowania tabeli w formularzu JSF

na podstawie

<https://docs.oracle.com/javase/7/JEETT.pdf>

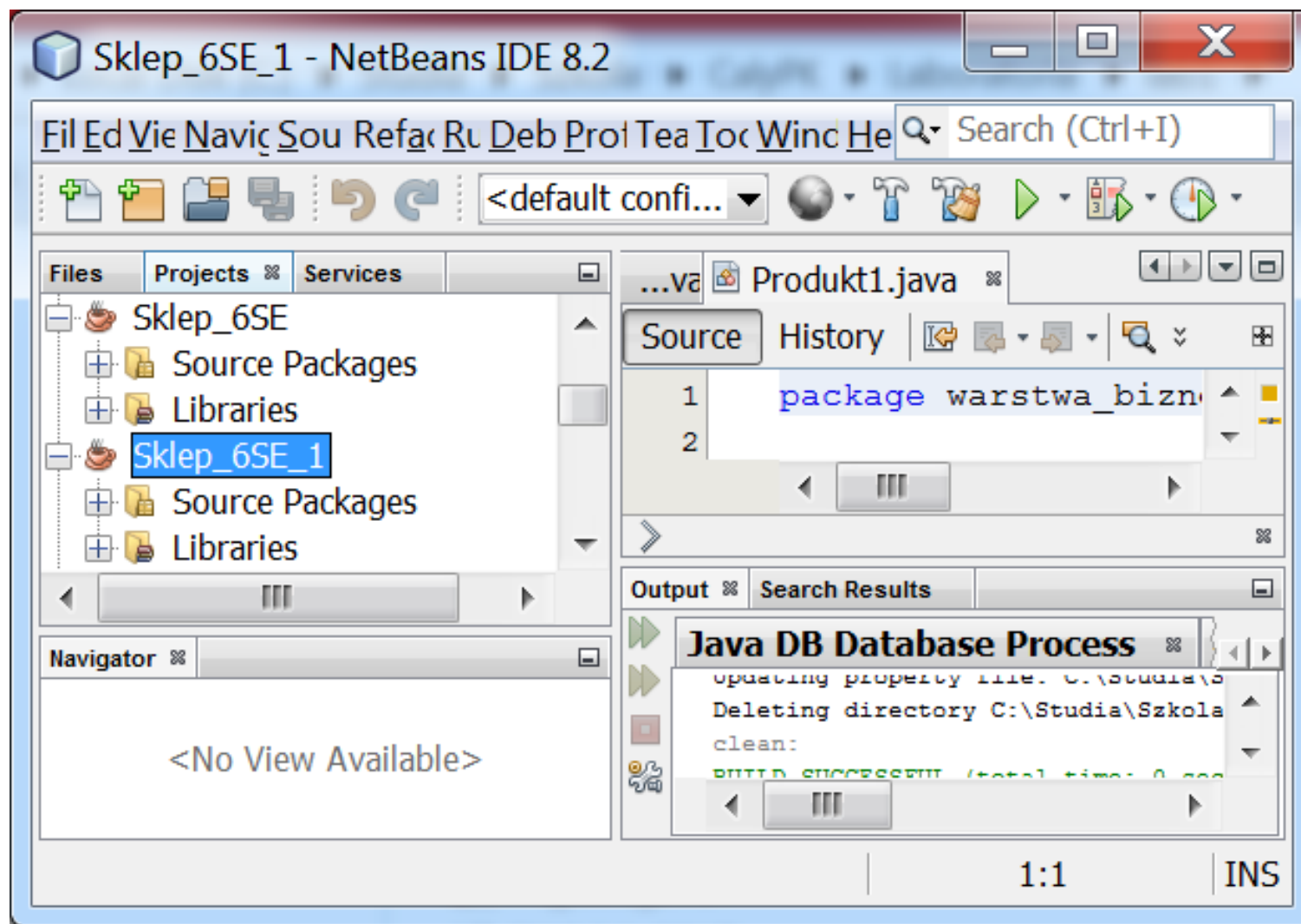
Programowanie komponentowe Lab3

1. Dodanie obiektu transferowego – w wyniku refaktoryzacji kodu aplikacji z lab2.

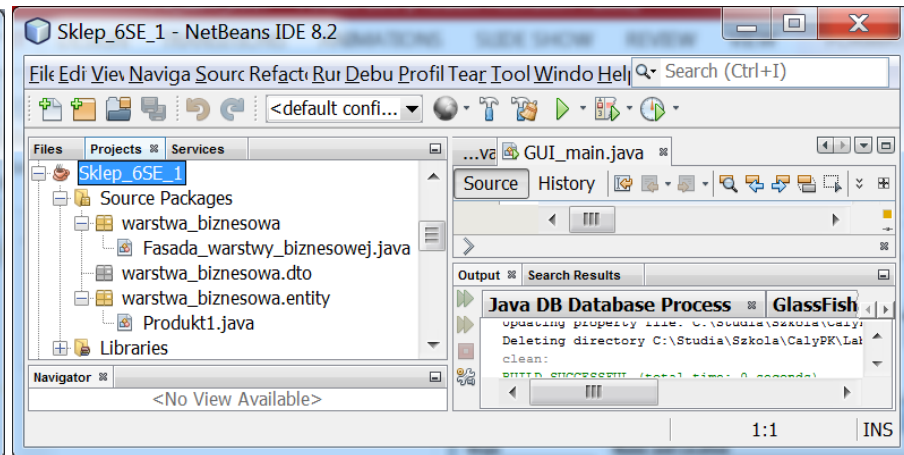
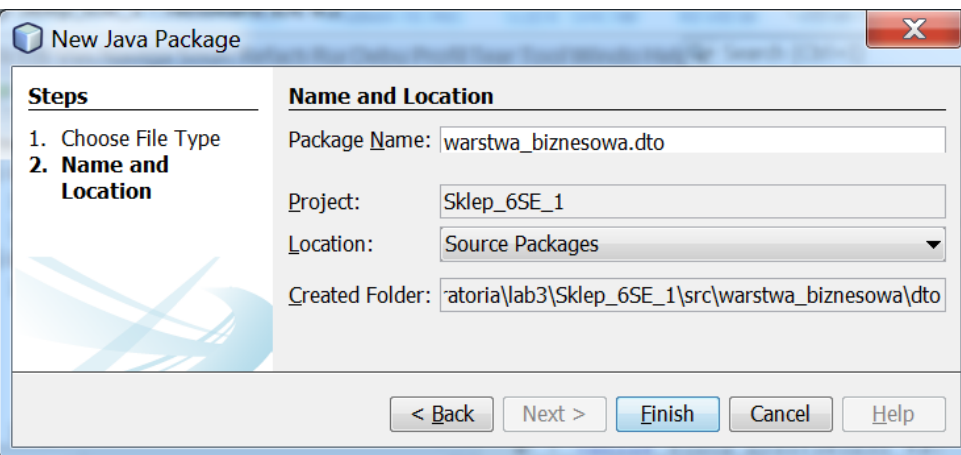
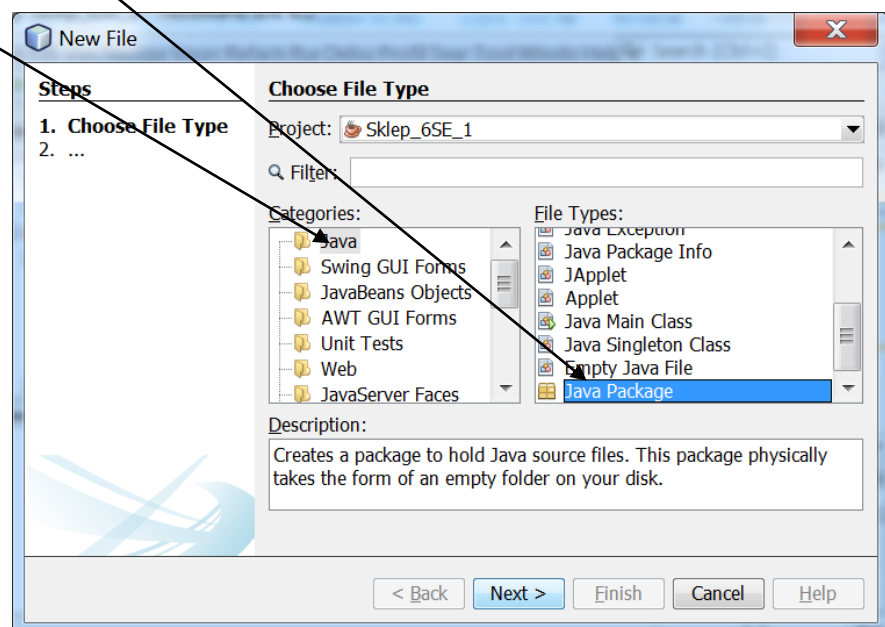
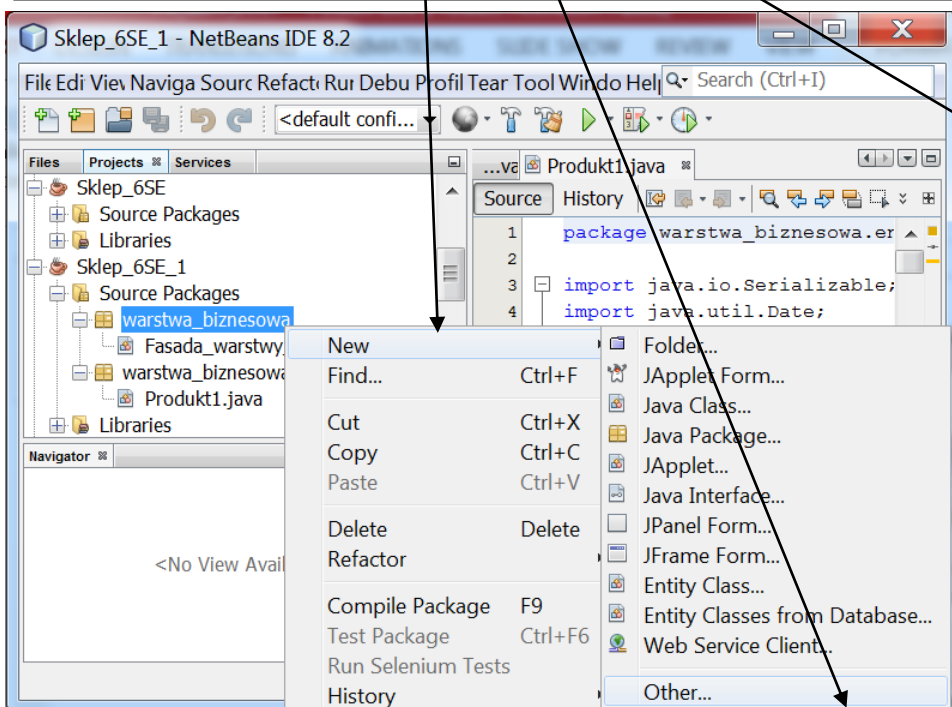
Materiał został zaprezentowany na wykładzie 6 z Technologii Internetowych (str. 12-42):

http://zofia.kruckiewicz.staff.iiar.pwr.wroc.pl/wyklady/ti/TINT_6.pdf

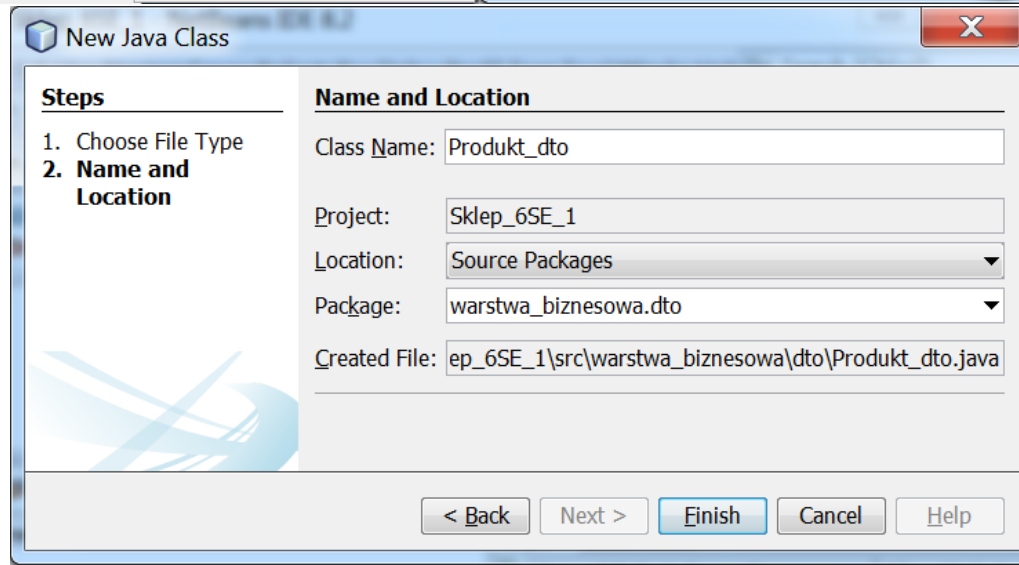
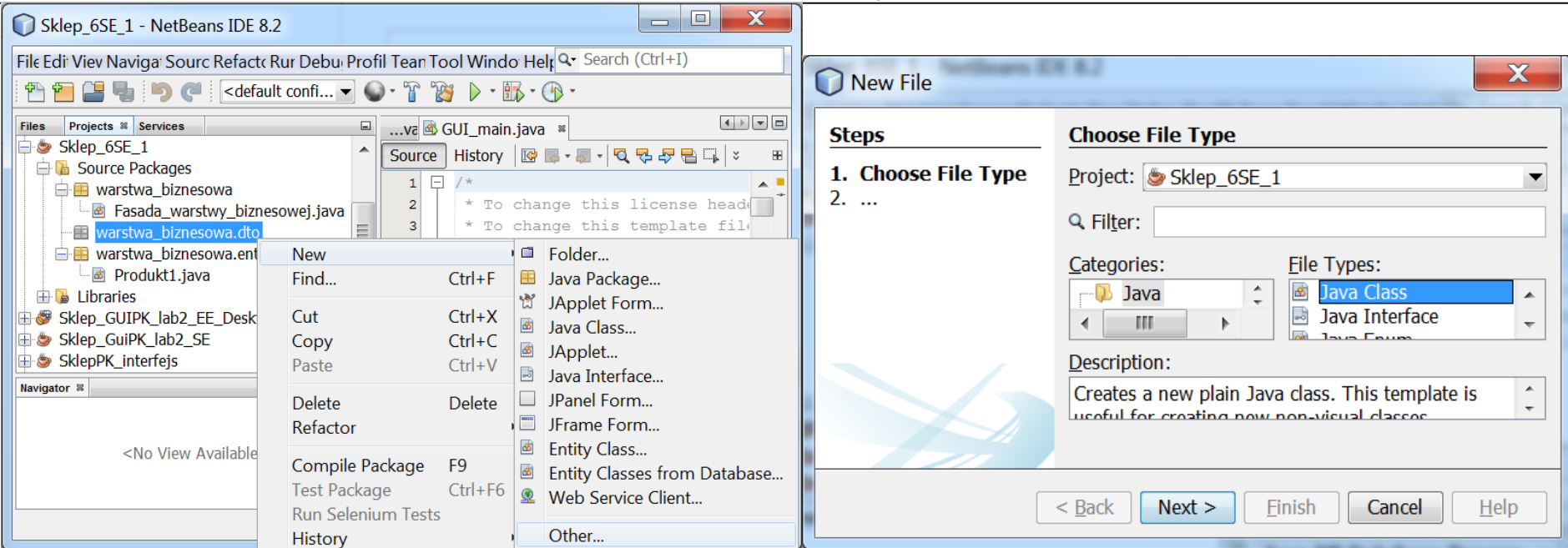
1.1. Wykonanie kopii programu **Sklep_6SE** jako **Sklep_6SE_1** wykonanego podczas lab2 (po zakończeniu kopiowania należy zamknąć program źródłowy)



1.1. Należy wykonać w projekcie **Sklep_6SE_1** nowy pakiet **warstwa_biznesowa.dto**: w okienku zakładki **Projects**, po kliknięciu prawym klawiszem na pakiet **warstwa_biznesowa**, wybrać kolejno **New/Other/Java/Java Package**. W polu **Package Name** uzupełnić nazwę o przyrostek: **.dto** i kliknąć **Finish**.

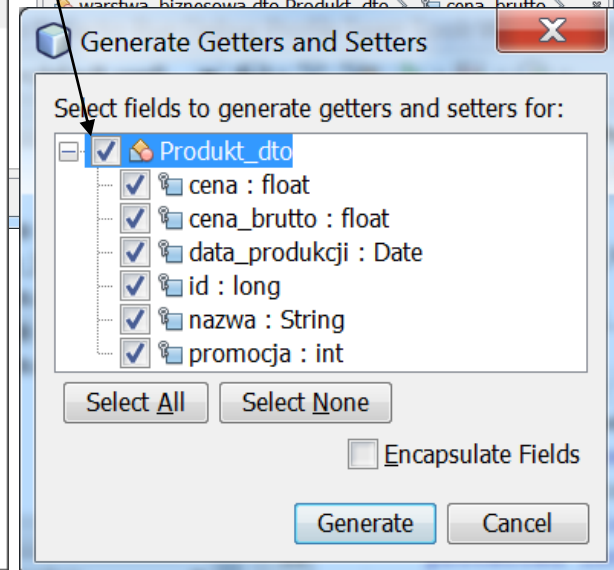
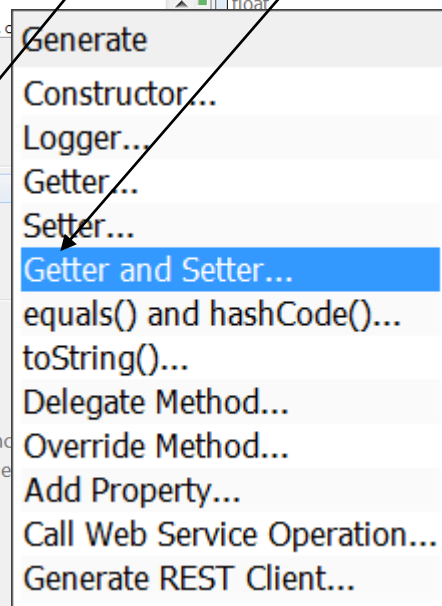
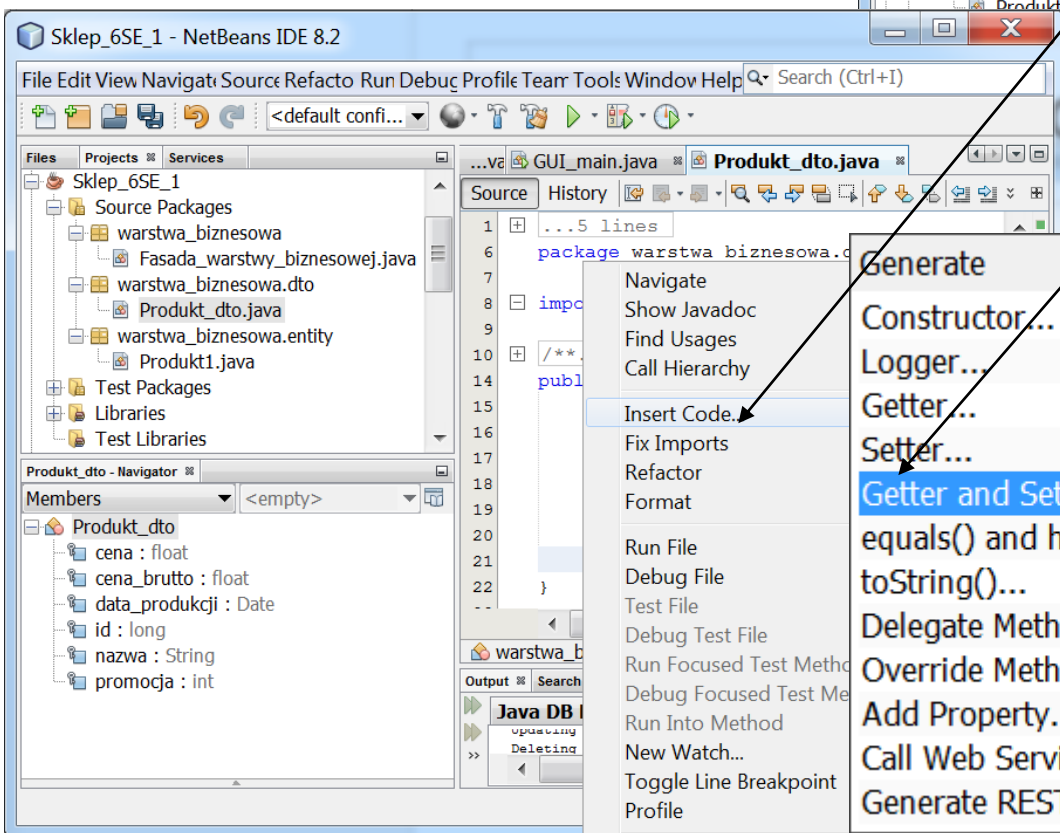
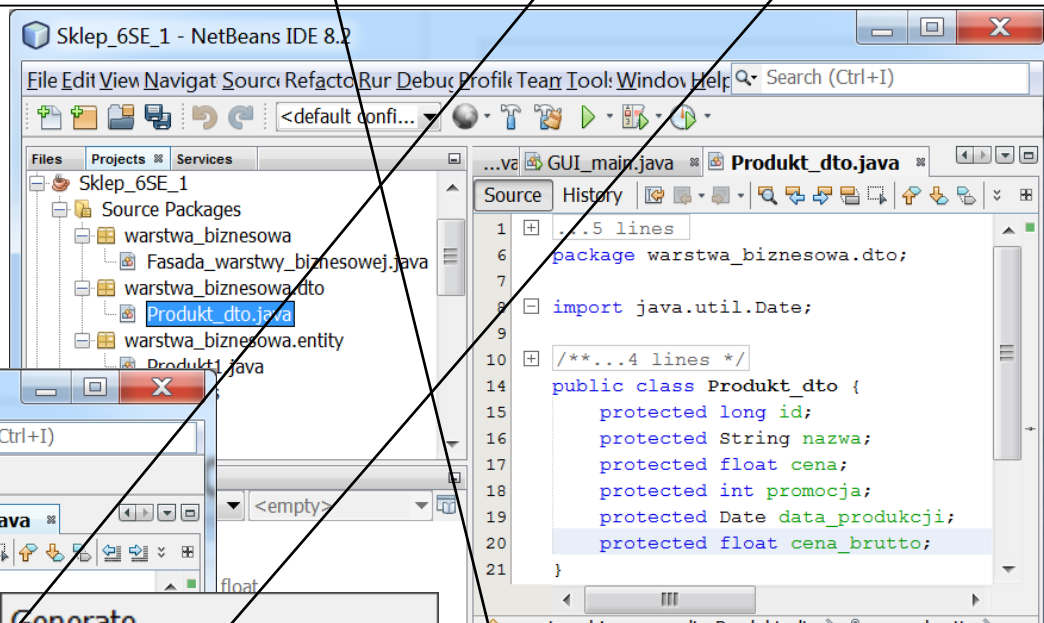


1.1. cd. W nowym pakiecie należy dodać klasę **Produkt_dto**, pełniącą rolę typu obiektu transferowego: po kliknięciu prawym klawiszem na pakiet **warstwa_biznesowa.dto** w okienku zakładki **Projects**, wybrać kolejno **New/Other/Java/Java Class**. Wpisać w polu **Class Name**: **Produkt_dto** i kliknąć **Finish**.



1.1. cd. W klasie **Produkt_dto** należy dodać atrybuty podane poniżej, z lewej strony (oczywiście, w przypadku własnej wersji obiektu typu Produkt1 należy uzupełnić zestaw tych atrybutów). Następnie, po kliknięciu prawym klawiszem na powierzchnię klasy w oknie edytora, należy wybrać kolejno pozycje **Insert Code.. /Getter and Setter** i w kolejnym oknie zaznaczyć **Produkt_dto** i kliknąć na **Generate**.

protected long id;
protected String nazwa;
protected float cena;
protected int promocja;
protected Date data_produkcji;
protected float cena_brutto;




```
public Produkt_dto(String [] dane, Date data)
{
    nazwa=dane[0];
    cena=Float.parseFloat(dane[1]);
    promocja=Integer.parseInt(dane[2]);
    data_produkcji=data;
}
```

1.1. cd Należy dodać konstruktor, podany z lewej strony, do kodu klasy **Produkt_dto (wersja podstawowa)** oraz **serializację** (następny slajd)

Sklep_6SE_1 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Files Services ...va Produkt_dto.java lista_produkow.xhtml Jodaj_produk2.xhtml Managed_produk...

Source History

```
6 package warstwa_biznesowa.dto;
7 import java.io.Serializable;
8 import java.util.Date;
9 /**...4 lines */
10 public class Produkt_dto implements Serializable{
11     protected long id;
12     protected String nazwa;
13     protected float cena;
14     protected int promocja;
15     protected Date data_produkcji;
16     protected float cena_brutto;
17     public Produkt_dto() {}
18     public Produkt_dto(String [] dane, Date data)
19     {
20         nazwa=dane[0];
21         cena=Float.parseFloat(dane[1]);
22         promocja=Integer.parseInt(dane[2]);
23         data_produkcji=data;
24     }
25     public long getId() { return id; }
26     public void setId(long id) { this.id = id; }
27     public String getNazwa() { return nazwa; }
28     public void setNazwa(String nazwa) { this.nazwa = nazwa; }
29     public float getCena() { return cena; }
30     public void setCena(float cena) { this.cena = cena; }
31     public int getPromocja() { return promocja; }
32     public void setPromocja(int promocja) { this.promocja = promocja; }
33     public Date getData_produkcji() { return data_produkcji; }
34     public void setData_produkcji(Date data_produkcji) { this.data_produkcji = data_produkcji; }
35     public float getCena_brutto() { return cena_brutto; }
36     public void setCena_brutto(float cena_brutto) { this.cena_brutto = cena_brutto; }
37 }
38
39
40
```

Produkt_dto - Navigator

Members

- Produkt_dto :: Serializable
- Produkt_dto()
- Produkt_dto(String[] dane, Date data)
- getCena(): float
- getCena_brutto(): float
- getData_produkcji(): Date
- getId(): long
- getNazwa(): String
- getPromocja(): int
- setCena(float cena)
- setCena_brutto(float cena_brutto)
- setData_produkcji(Date data_produkcji)
- setId(long id)
- setNazwa(String nazwa)
- setPromocja(int promocja)
- cena: float
- cena_brutto: float
- data_produkcji: Date
- id: long
- nazwa: String
- promocja: int

warstwa_biznesowa.dto.Produkt_dto > Produkt_dto >

Output

Java DB Database Process GlassFish Server 4.1.1 Sklep_GUIPK_lab3_EE_Desktop (run)

20:53 INS

1.1. cd Należy również zachować **konstruktor domyślny**, który teraz musi być jawnie zdefiniowany

1.1. cd Należy dodać możliwość serializacji obiektu typu **Produkt_dto**
– ponieważ tylko obiekty serializowane mogą być przesyłane przez sieć
z wykorzystaniem kontenerów

1) Informacja o kontenerach Java EE: strony 9-10 w pliku z wykładu 1:
http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/javapk/PK_1.pdf

2) Uzupełnienie definicji klasy **Produkt_dto**:

```
package warstwa_biznesowa.dto;
```

```
import java.io.Serializable;
```

```
import java.util.Date;
```

```
public class Produkt_dto implements Serializable{
```


1.2. W klasie **Fasada_warstwy_biznesowej**, w pakiecie **warstwa_biznesowa** projektu **Sklep_6SE_1** zmodyfikowac kody podanych ponizej metod (metoda **dodaj_produk**t pozostaje bez zmian), wykorzystujac teraz obiekt transferowy typu **Produkt_dto** (wyklad http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/ti_/TINT_6.pdf)

```
public void utworz_produk(Produkt_dto produkt_dto) {
    Produkt1 produkt = new Produkt1();
    klucz++;
    produkt.setId(new Long(klucz));
    produkt.setNazwa(produkt_dto.getNazwa());
    produkt.setCena(produkt_dto.getCena());
    produkt.setPromocja(produkt_dto.getPromocja());
    produkt.setData_produkcji(produkt_dto.getData_produkcji());
    dodaj_produk(produkt);
}

protected void dodaj_produk(Produkt1 produkt) {
    if (!produkty.contains(produkt)) {
        produkty.add(produkt);
        stan = true;
    } else
        stan = false;
}
```

1.2. cd. Również należy zmodyfikować i dodać metody tworzące modele danych widoków np formularzy **dodaj_produk2.xhtml**, **rezultat2.xhtml**, **lista_produkow.xhtml** oraz formularzy aplikacji desktopowej **SE i EE**.

```
public Produkt_dto dane_produktu() {  
    if (stan) {  
        Produkt1 produkt = produkty.get(produkty.size() - 1);  
        return produkt_transfer(produkt);  
    }  
    return null;  
}
```

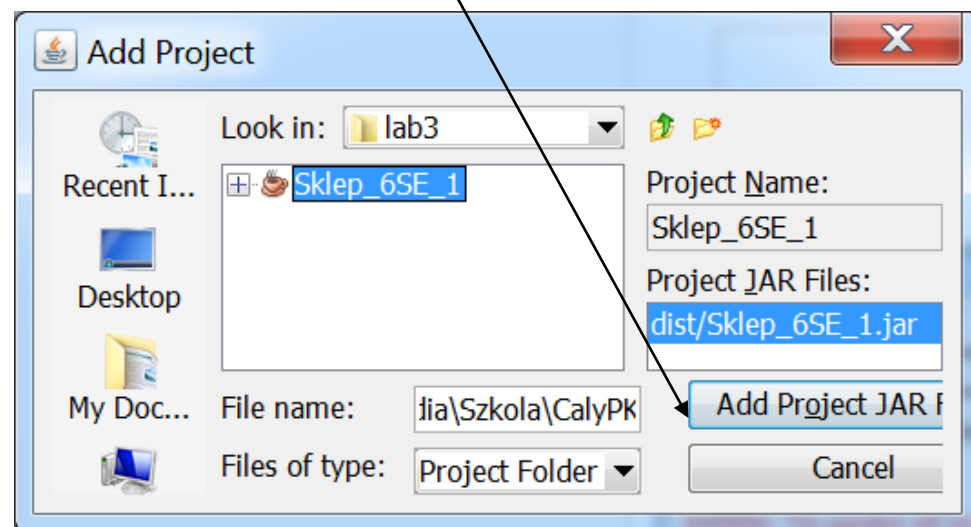
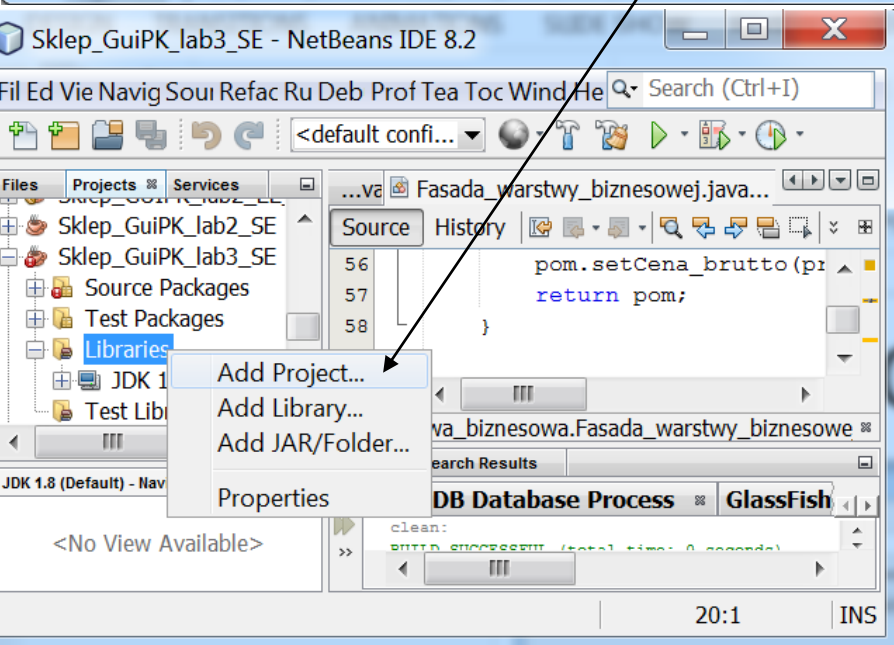
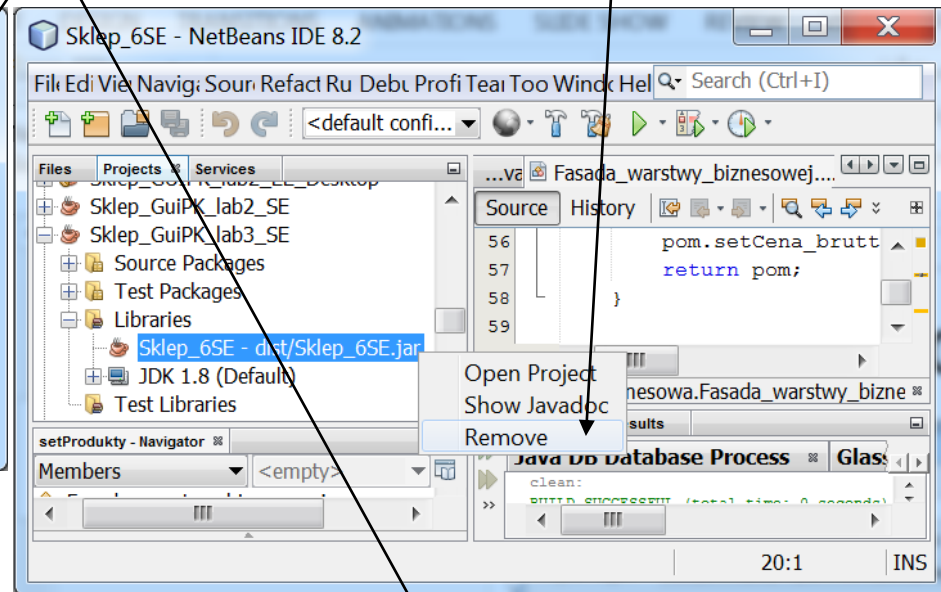
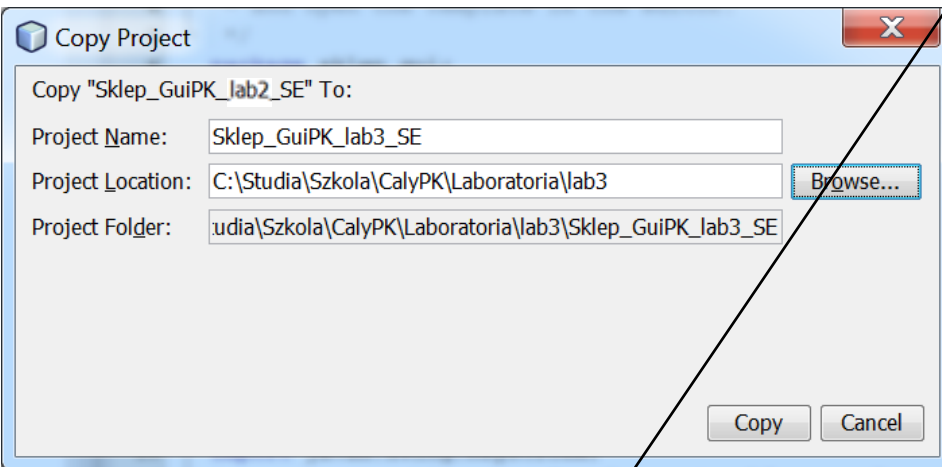
```
public Produkt_dto produkt_transfer(Produkt1 produkt) {  
    Produkt_dto pom = new Produkt_dto();  
    pom.setId(produkt.getId());  
    pom.setNazwa(produkt.getNazwa());  
    pom.setCena(produkt.getCena());  
    pom.setPromocja(produkt.getPromocja());  
    pom.setData_produkcji(produkt.getData_produkcji());  
    pom.setCena_brutto(produkt.cena_brutto());  
    return pom;  
}
```

1.2. cd Również należy zmodyfikować metody tworzące modele danych widoków np formularzy `dodaj_produk2.xhtml`, `rezultat2.xhtml`, `lista_produkow.xhtml` oraz formularzy aplikacji desktopowej SE i EE. Należy pozostawić dotychczasową metodę `items`.

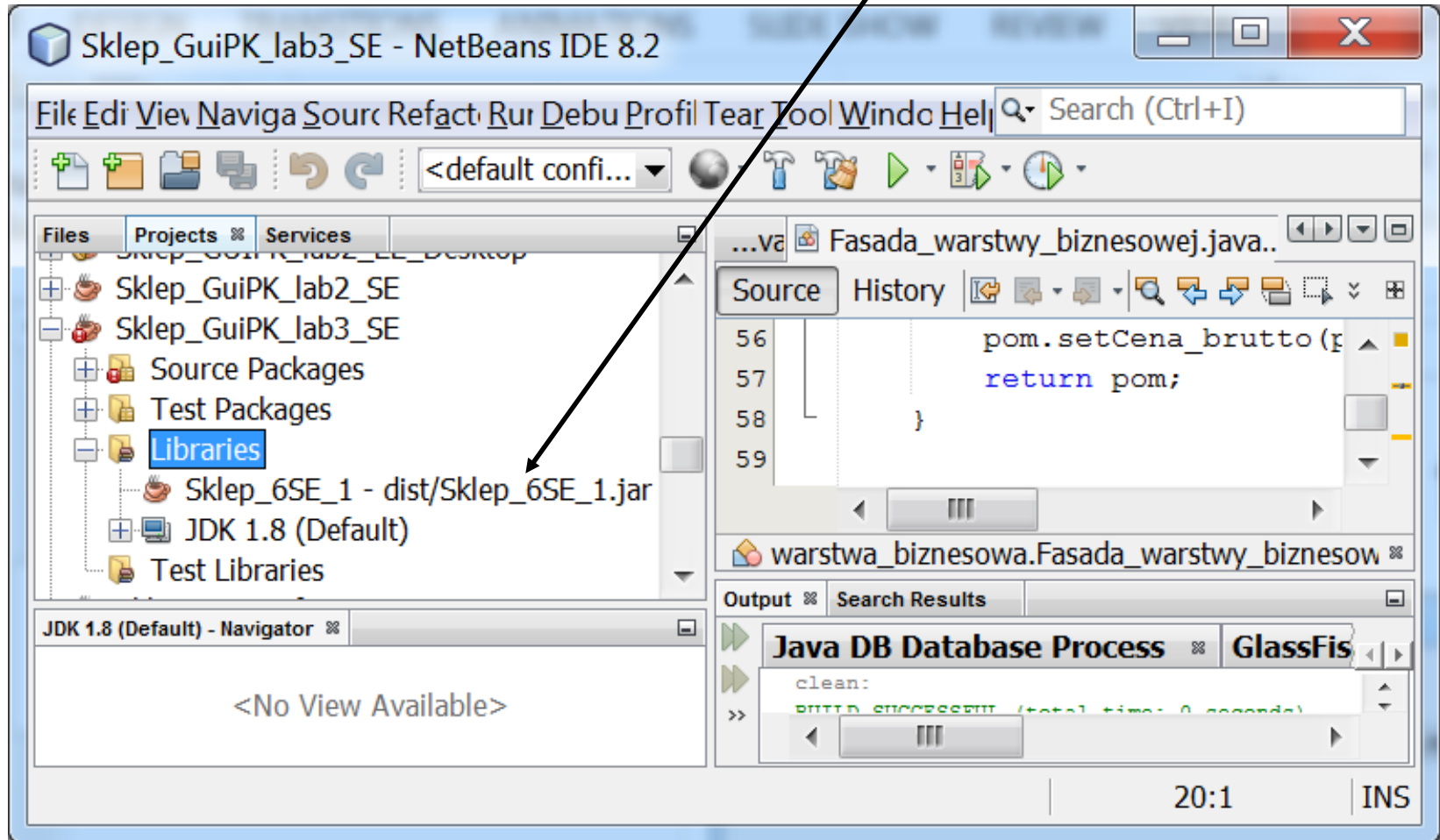
```
public ArrayList<Produkt_dto> items_() {  
    ArrayList<Produkt_dto> dane = new ArrayList();  
    for (Produkt1 produkt : produkty)  
        dane.add(produkt_transfer(produkt));  
    return dane;  
}  
  
public ArrayList<ArrayList<String>> items() {  
    ArrayList<ArrayList<String>> dane = new ArrayList();  
    for (Produkt1 p : produkty) {  
        ArrayList<String> wiersz = new ArrayList();  
        wiersz.add(p.getId().toString());  
        wiersz.add(p.getNazwa());  
        wiersz.add("" + p.getCena());  
        wiersz.add("" + p.getPromocja());  
        wiersz.add(p.getData_produkcji().toString());  
        wiersz.add("" + p.cena_brutto());  
        dane.add(wiersz);  
    }  
    return dane;  
}
```

Nowa metoda `items_` tworząca model widoku `dataTable` na stronie `lista_produkow.xhtml`

1.3. W celu modyfikacji kodu desktopowego klienta EE należy najpierw wykonać kopię programu **Sklep_GuiPK_lab2_SE** jako **Sklep_GuiPK_lab3_SE**. Następnie, należy w katalogu **Libraries** tego projektu usunąć połączenie z dotychczasowym projektem **Sklep_6SE** (wybrać pozycję **Remove**) i zastąpić go projektem **Sklep_6SE_1** – poniżej ilustracja przebiegu tych czynności.



1.3. cd. Wynik połączenia z nowym projektem **Sklep_6SE_1** z definicjami aktualnego kodu klasy **Produkt_dto** i zmodyfikowanego kodu logiki biznesowej.



1.3. cd. Należy zmodyfikować kod metody **actionPerformed** obsługującej zdarzenie kliknięcia na przycisk **Dodaj produkt** w pliku **Produkt_form**. Następnie, należy uruchomić program **Sklep_GuiPK_lab3_SE** w celu sprawdzenia poprawności jego działania – przed wykorzystaniem kodu tego projektu do utworzenia kodu aplikacji typu **Enterprise ApplicationClient**.

@Override

```
public void actionPerformed(ActionEvent evt) //obsługa zdarzenia kliknięcia na przycisk "Dodaj produkt"
{ String[] dane = form_produkt(); //utworzenie tablicy z danymi produktu: nazwa, cena, promocja
  if (dane == null)
    return;
  Date data_ = data(); //utworzenie daty
  if (data_ == null)
    return;
  Produkt_dto produkt = new Produkt_dto(dane,data_); // wywołanie metody
  GUI_main.getFacade().utworz_produkt(produkt); //logiki biznesowej tworzącej obiekt typu Produkt1
}
```

The screenshot shows two windows of the 'MenuDemo' application. The left window is the 'Dodaj produkt' form, and the right window is the main application window displaying a table of products.

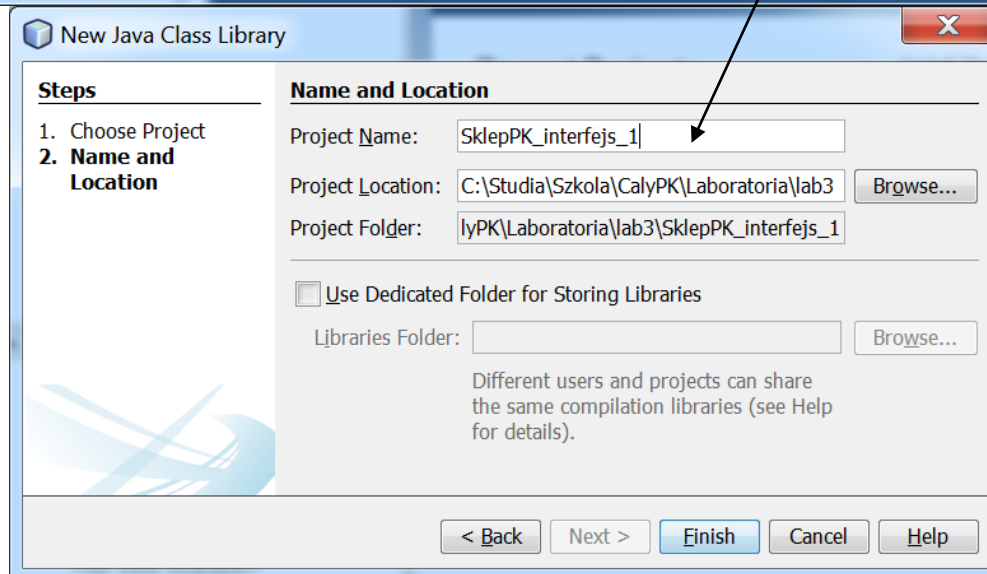
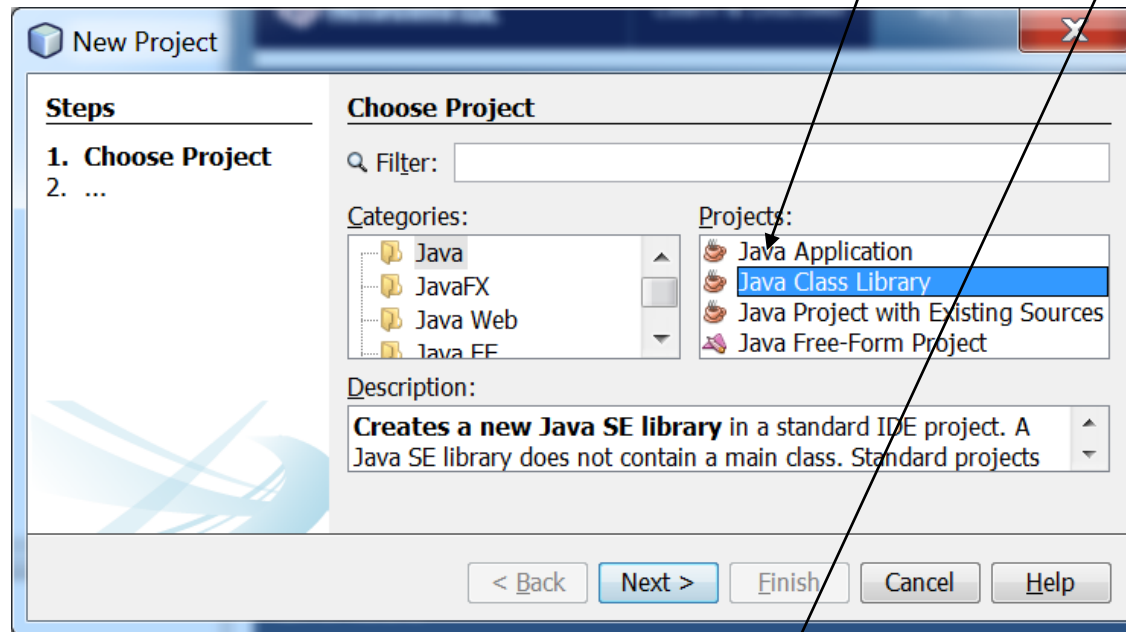
Form Data:

Nazwa	Cena	Promocja	Data
Produkt1	123	12	26-03-2017

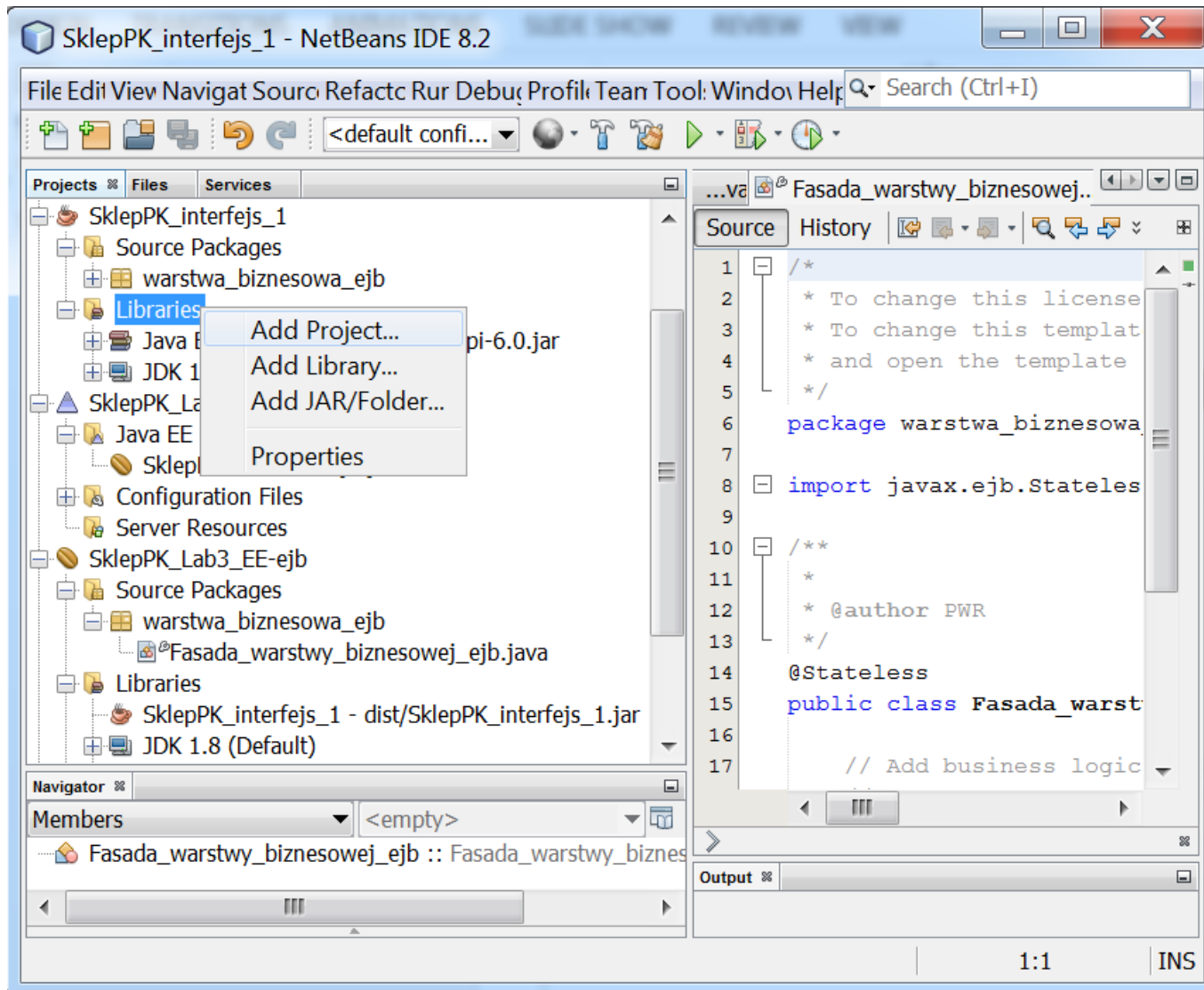
Product List Table:

Id produktu	Nazwa	Cena	Promocja	Data	Cena brutto
1	Produkt1	123.0	12	Sun Mar 26 03:19:25 CEST 2017	108.24

1.4. Następnie, należy założyć nowy projekt **Sklep_interfejs_1** (Files/New Project/Java/Java Class Library)



1.4. cd. Do projektu **SklepPK_interfejs_1** należy dodać w katalogu **Libraries** projekt **Sklep_6SE_1** (podobnie jak na str. 12) – jest to przygotowanie do definicji przyszłego interfejsu zdalnego dostępu do logiki biznesowej.

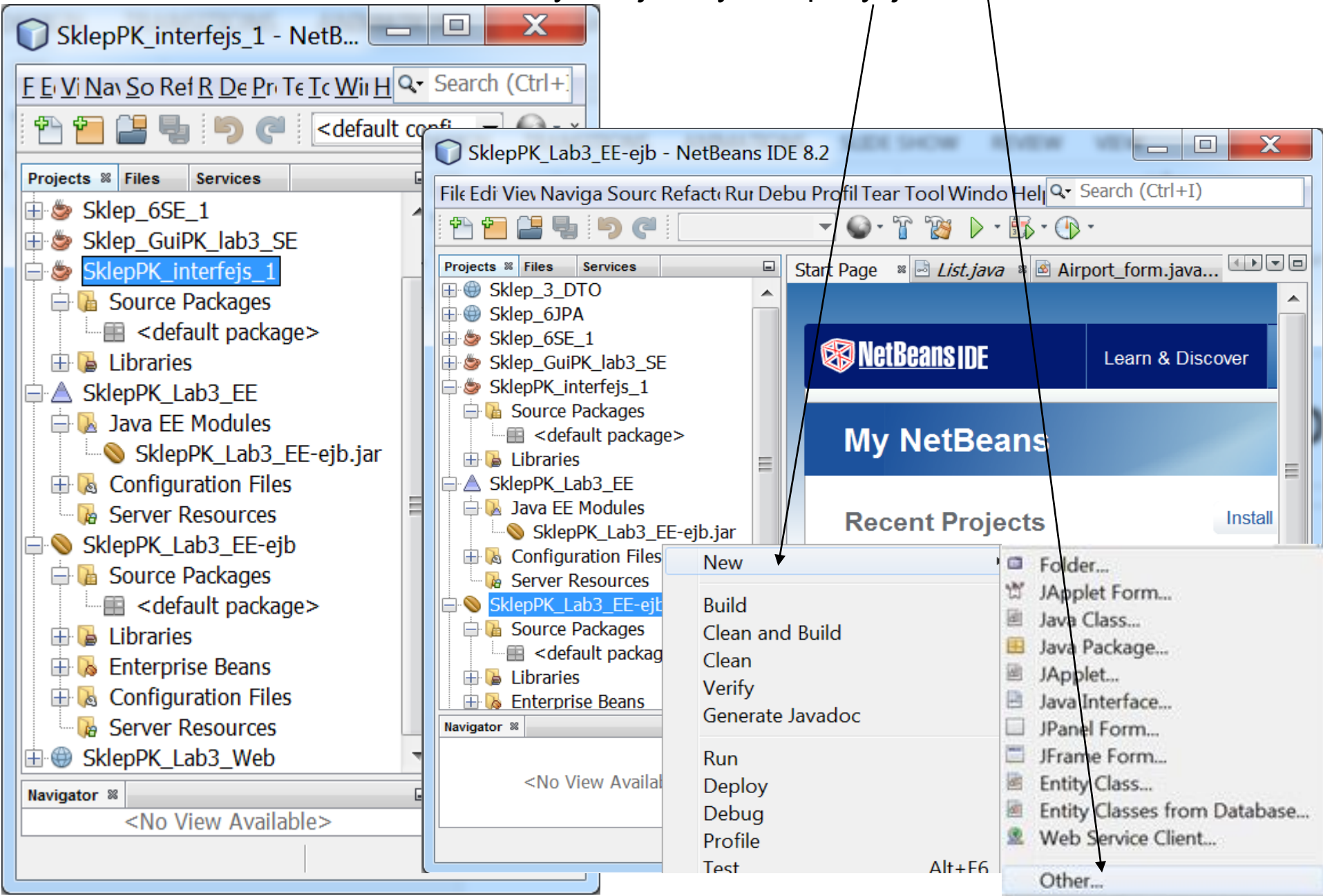


1.5. Należy wykonać główny projekt **Java EE** typu **Enterprise Application** (**Files/New Project/ Java EE/ Enterprise Application**) o nazwie **SklepPK_Lab3_EE**. Należy dodać jedynie moduł EJB **SklepPK_Lab3_EE-ejb** i nacisnąć **Finish**.

The image displays three sequential screenshots of the Eclipse IDE's project creation wizards, with arrows indicating the flow of the process:

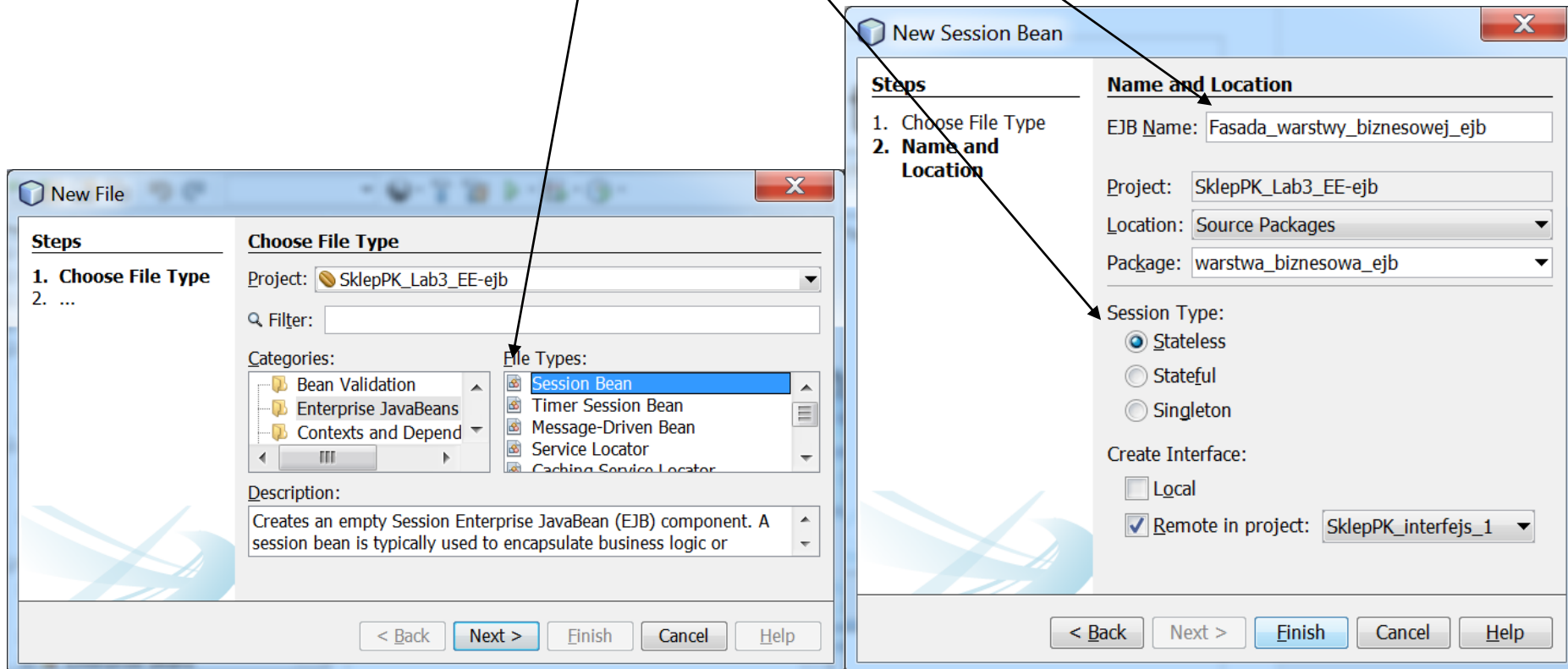
- Top Left Screenshot: 'New Project' Wizard**
 - Steps:** 1. Choose Project, 2. ...
 - Choose Project:** The 'Categories' list on the left has 'Java EE' selected. The 'Projects' list on the right has 'Enterprise Application' selected.
 - Description:** 'Creates a new enterprise application in a standard project. You can also create an EJB module project and Web application project in the enterprise application. A standard'.
 - Buttons:** < Back, Next >, Finish, Cancel, Help.
- Top Right Screenshot: 'New Enterprise Application' Wizard (Step 2)**
 - Steps:** 1. Choose Project, 2. Name and Location, 3. Server and Settings.
 - Name and Location:**
 - Project Name: SklepPK_Lab3_EE
 - Project Location: C:\Studia\Szkola\CalyPK\Laboratoria\lab3
 - Project Folder: Szkola\CalyPK\Laboratoria\lab3\SklepPK_Lab3_EE
 - Use Dedicated Folder for Storing Libraries
 - Libraries Folder: [empty]
 - Text:** 'Different users and projects can share the same compilation libraries (see Help for details).'
 - Buttons:** < Back, Next >, Finish, Cancel, Help.
- Bottom Screenshot: 'New Enterprise Application' Wizard (Step 3)**
 - Steps:** 1. Choose Project, 2. Name and Location, 3. Server and Settings.
 - Server and Settings:**
 - Server: GlassFish Server 4.1.1
 - Java EE Version: Java EE 7
 - Create EJB Module: SklepPK_Lab3_EE-ejb
 - Create Web Application Module: SklepPK_Lab3_EE-war
 - Buttons:** < Back, Next >, Finish, Cancel, Help.

1.5. cd. Należy do modułu **SklepPK_Lab3_EE-ejb** dodać komponent EJB typu **Session Bean**. Należy kolejno wybrać pozycje: **New/Other**



1.5. cd. Następnie, należy wybrać **Enterprise JavaBean/ Session Bean**.

Nowy komponent EJB typu **Session Bean** ma nadaną nazwę **Fasada_warstwy_biznesowej_ejb**. Podczas tworzenia wybrać następujące właściwości komponentu EJB: **Stateless, Remote** i wybrać z listy projekt **SklepPK_interfejs_1** do zdefiniowania interfejsu tworzonego komponentu typu SessionBean



1.5. cd. Należy zdefiniować **interfejs logiki biznesowej**, do które będą odwoływać się **aplikacje warstwy klienta: internetowa i desktopowa.**

```
package warstwa_biznesowa_ejb;
```

```
import java.util.ArrayList;
```

```
import javax.ejb.Remote;
```

```
import warstwa_biznesowa.dto.Produkt_dto;
```

@Remote

```
public interface Fasada_warstwy_biznesowej_ejbRemote {  
    public void utworz_produkt(Produkt_dto produkt_dto);  
    public Produkt_dto dane_produktu();  
    public ArrayList<ArrayList<String>> items();  
    public ArrayList<Produkt_dto> items_();  
}
```

1.5. cd. Rezultat

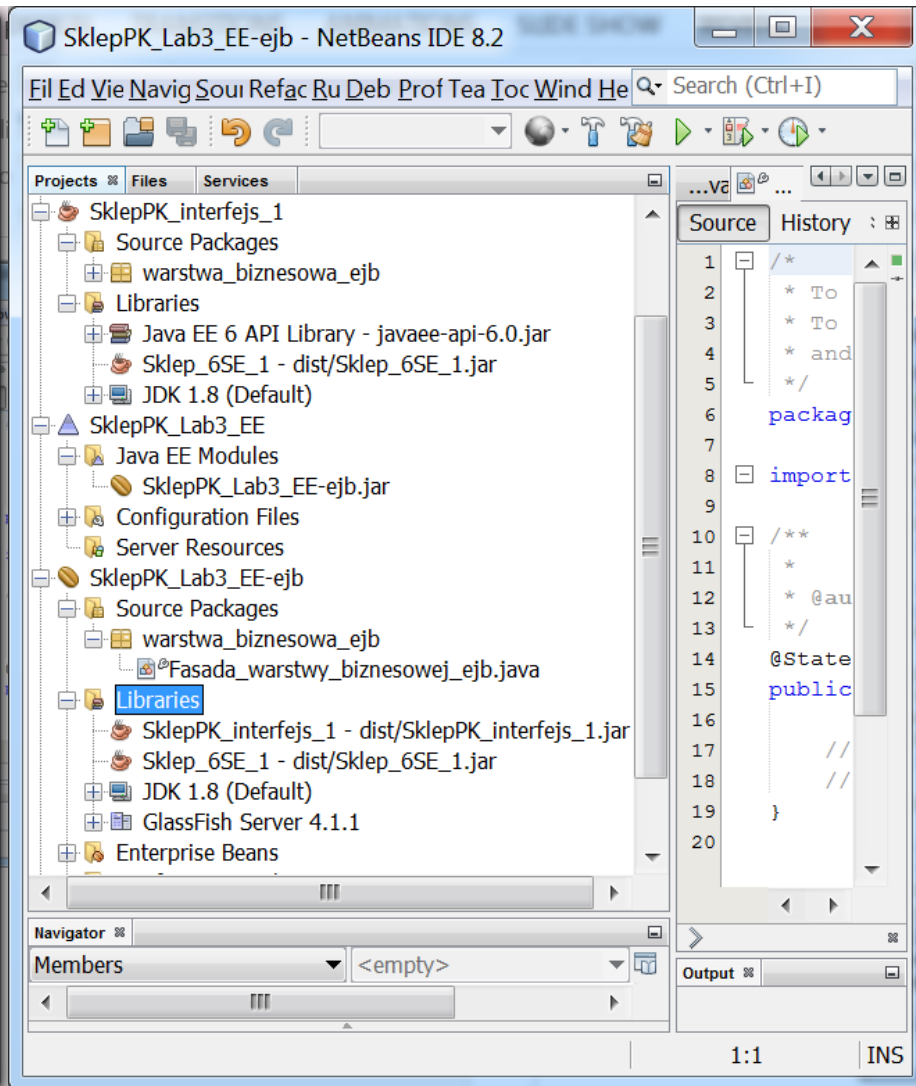
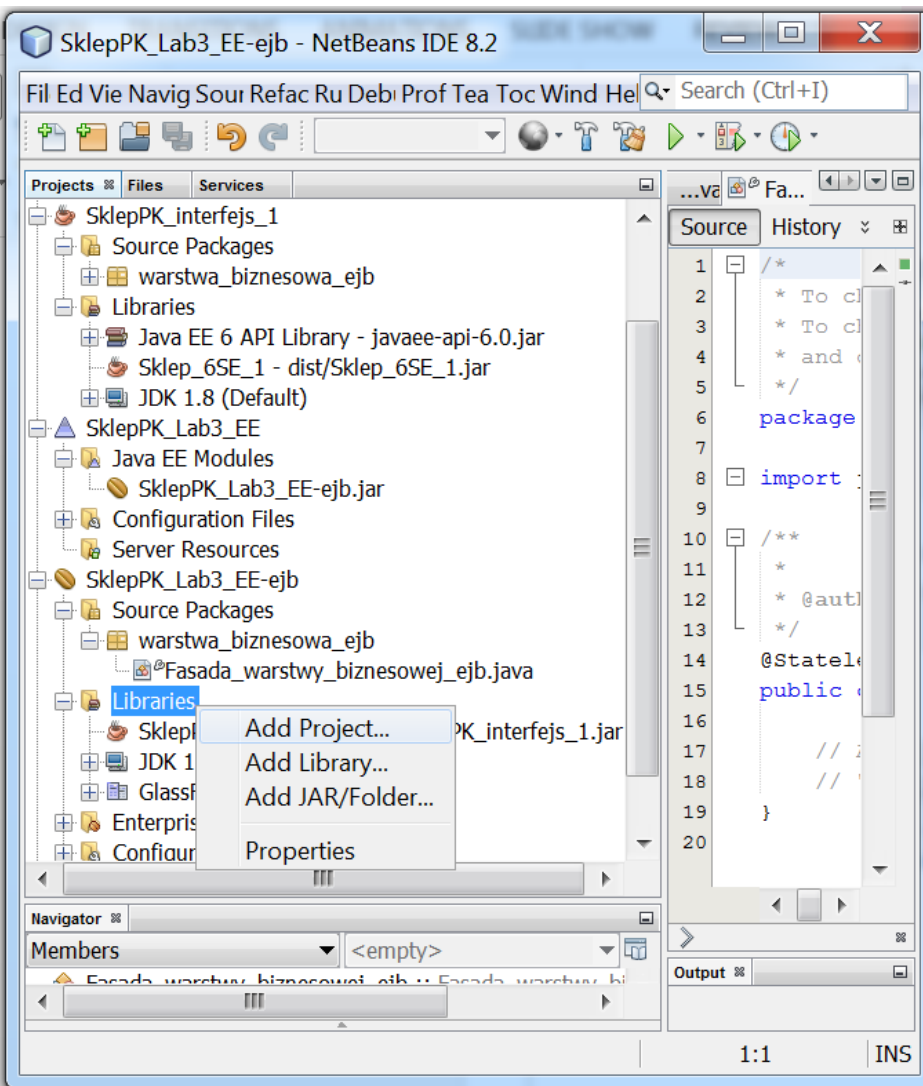
The screenshot shows the NetBeans IDE 8.2 interface. The main editor displays the source code for the file `Fasada_warstwy_biznesowej_ejbRemote.java`. The code is as follows:

```
1  ...5 lines
6  package warstwa_biznesowa_ejb;
7
8  import java.util.ArrayList;
9  import javax.ejb.Remote;
10 import warstwa_biznesowa.dto.Produkt_dto;
11
12 /**...4 lines */
16 @Remote
17
18 public interface Fasada_warstwy_biznesowej_ejbRemote {
19     public void utworz_produkt(Produkt_dto produkt_dto);
20     public Produkt_dto dane_produktu();
21     public ArrayList<ArrayList<String>> items();
22     public ArrayList<Produkt_dto> items_();
23 }
```

The `items_()` method is highlighted in the editor. The left sidebar shows the project structure for `SklepPK_interfejs_1`, with `Fasada_warstwy_biznesowej_ejbRemote.java` selected. The bottom panel shows the `Members` view for the `items_()` method, listing its return type `ArrayList<Produkt_dto>`.

21:44 | INS

1.5. cd. Do modułu **EJB SklepPK_Lab3_EE-ejb** należy dodać w katalogu **Libraries** projekt **Sklep_6SE_1** (podobnie jak na str. 12) – jest to przygotowanie do **implementacji** interfejsu zdalnego dostępu do logiki biznesowej.

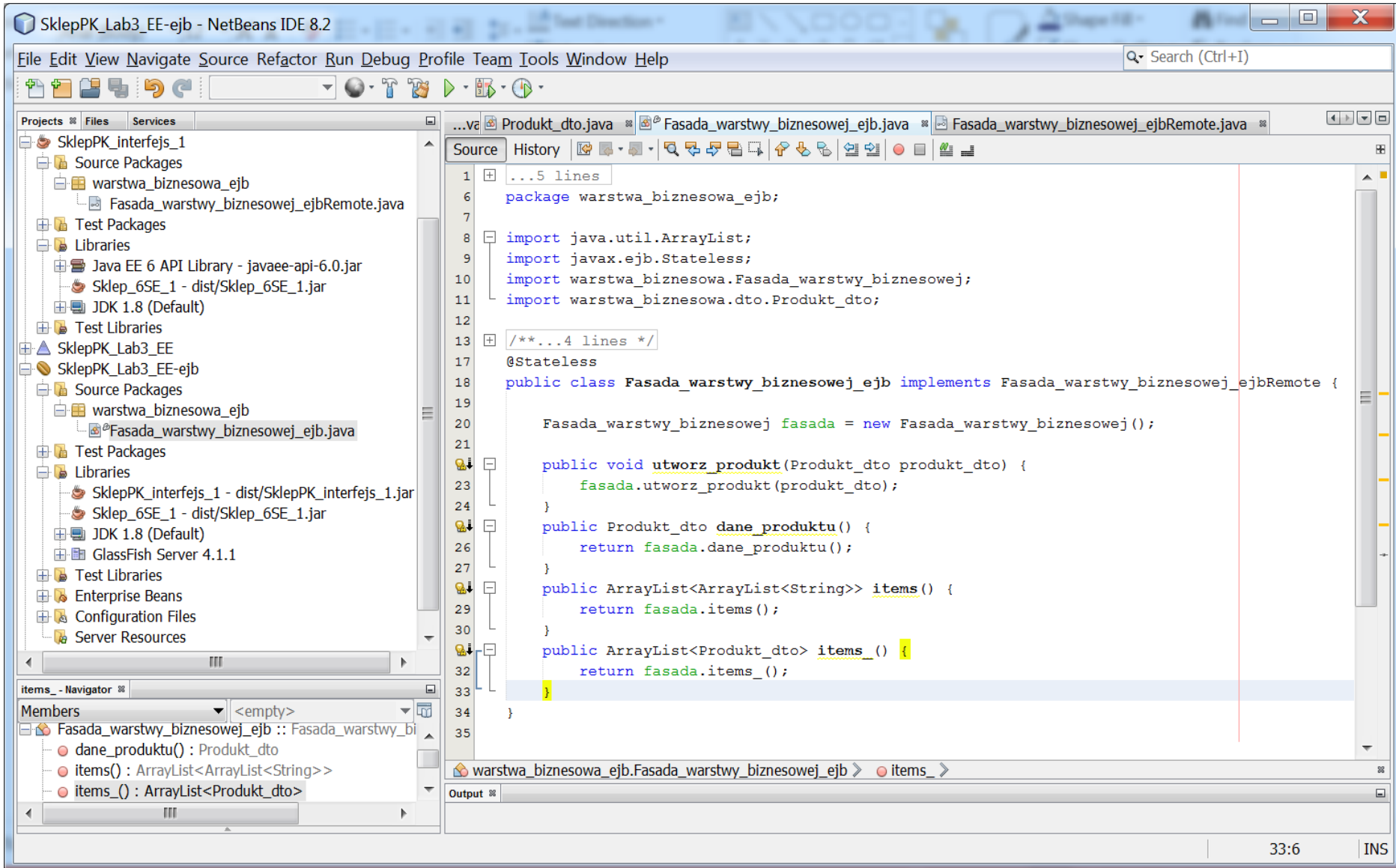


1.5. cd. W klasie komponentu EJB typu **Session Bean** o nazwie **Fasada_warstwy_biznesowej_ejb** należy zdefiniować metody zadeklarowane w interfejsie **Fasada_warstwy_biznesowej_ejbRemote**, opierające się na kodzie metod klasy **Fasada_warstwy_biznesowej** – klasa typu **Fasada_warstwy_biznesowej_ejb** jest „mostem sesyjnym” umożliwiającym wywołanie metod logiki biznesowej

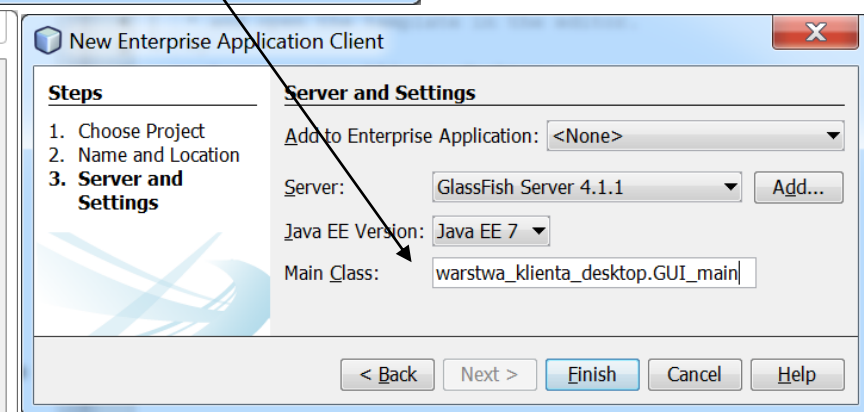
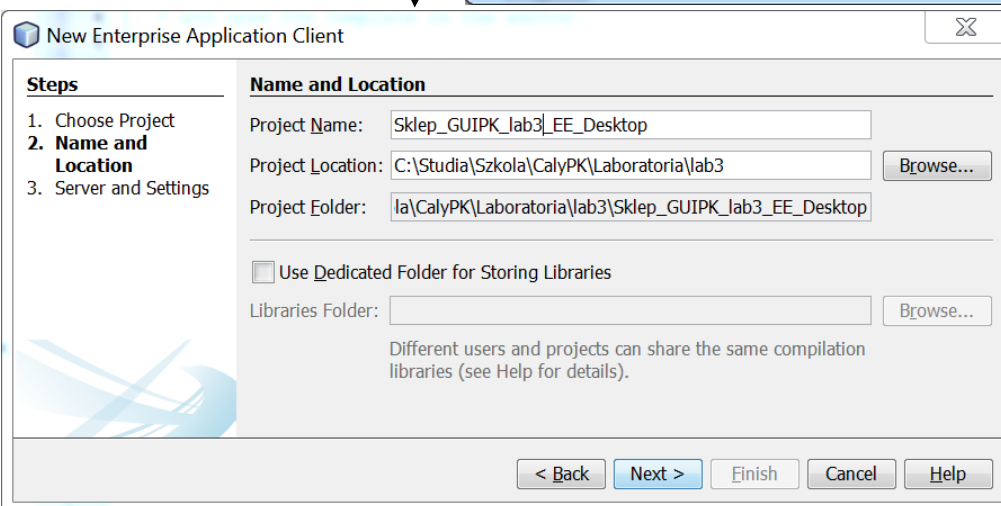
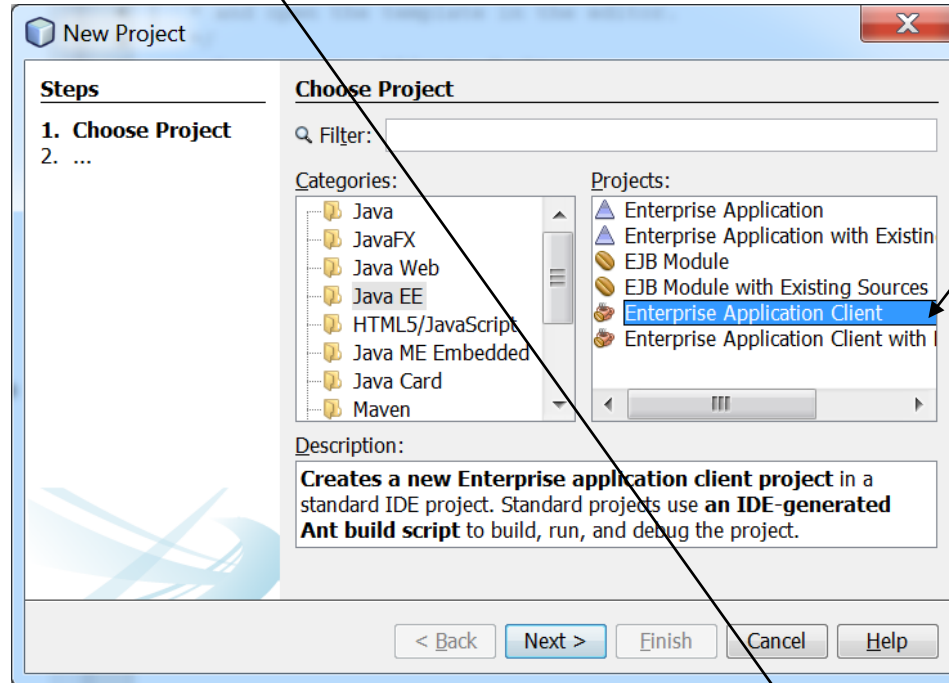
```
package warstwa_biznesowa_ejb;
import java.util.ArrayList;
import javax.ejb.Stateless;
import warstwa_biznesowa.Fasada_warstwy_biznesowej;
import warstwa_biznesowa.dto.Produkt_dto;
@Stateless
public class Fasada_warstwy_biznesowej_ejb implements Fasada_warstwy_biznesowej_ejbRemote {
    Fasada_warstwy_biznesowej fasada = new Fasada_warstwy_biznesowej();

    public void utworz_produkt(Produkt_dto produkt_dto) {
        fasada.utworz_produkt(produkt_dto);
    }
    public Produkt_dto dane_produktu() {
        return fasada.dane_produktu();
    }
    public ArrayList<ArrayList<String>> items() {
        return fasada.items();
    }
    public ArrayList<Produkt_dto> items_() {
        return fasada.items_();
    }
}
```

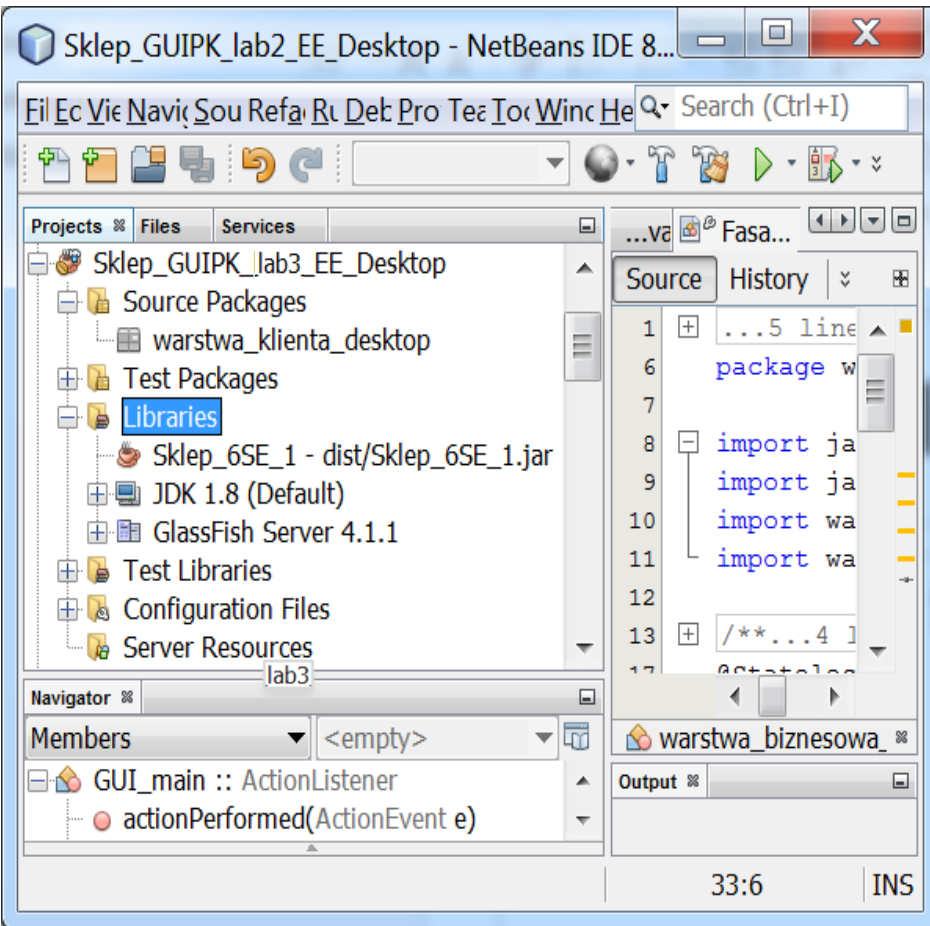
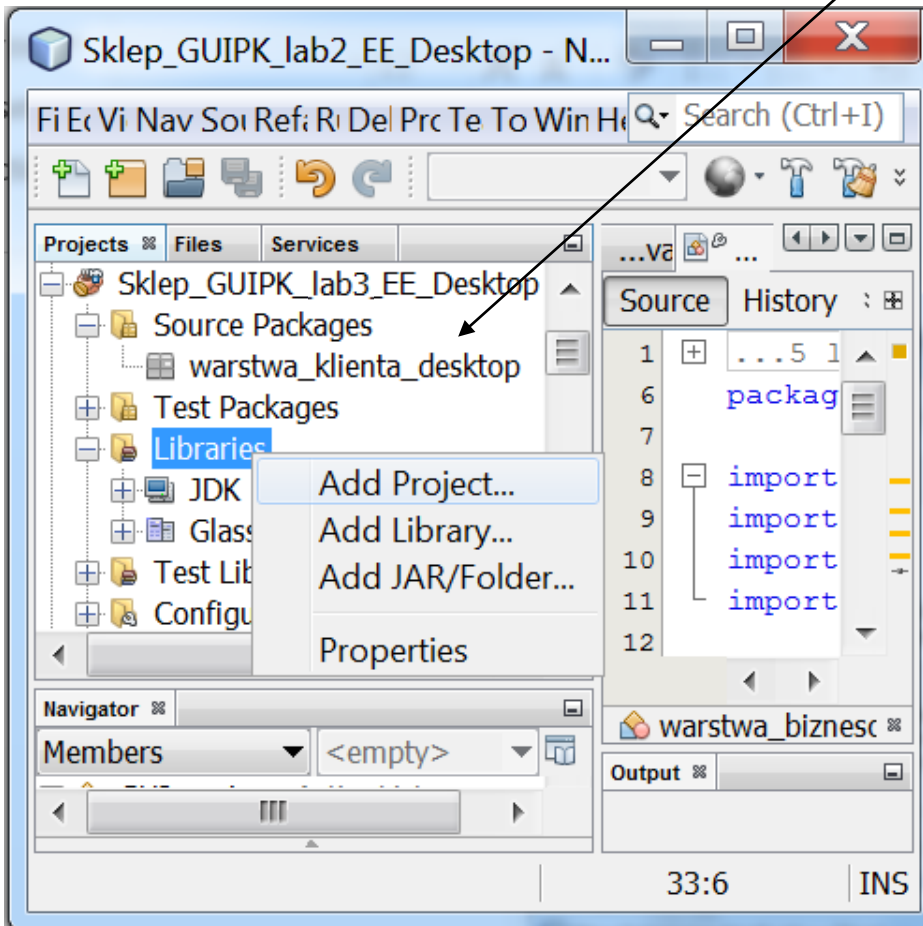
1.5. cd. Rezultat



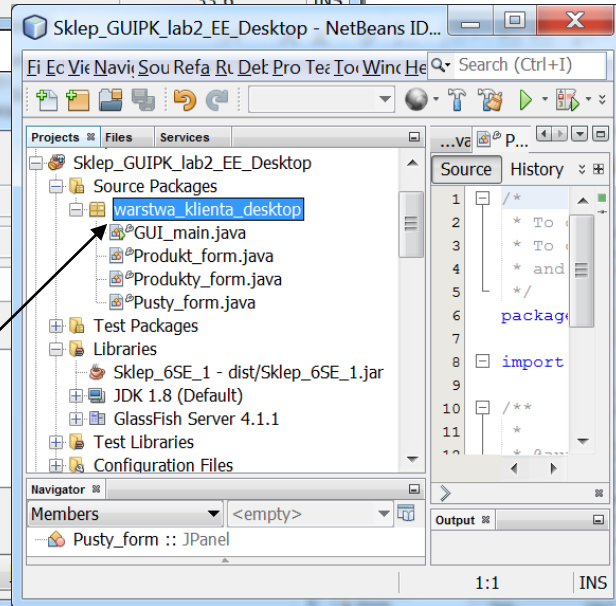
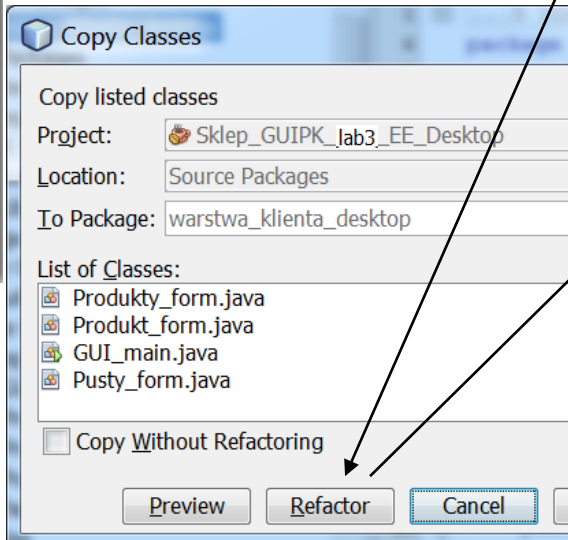
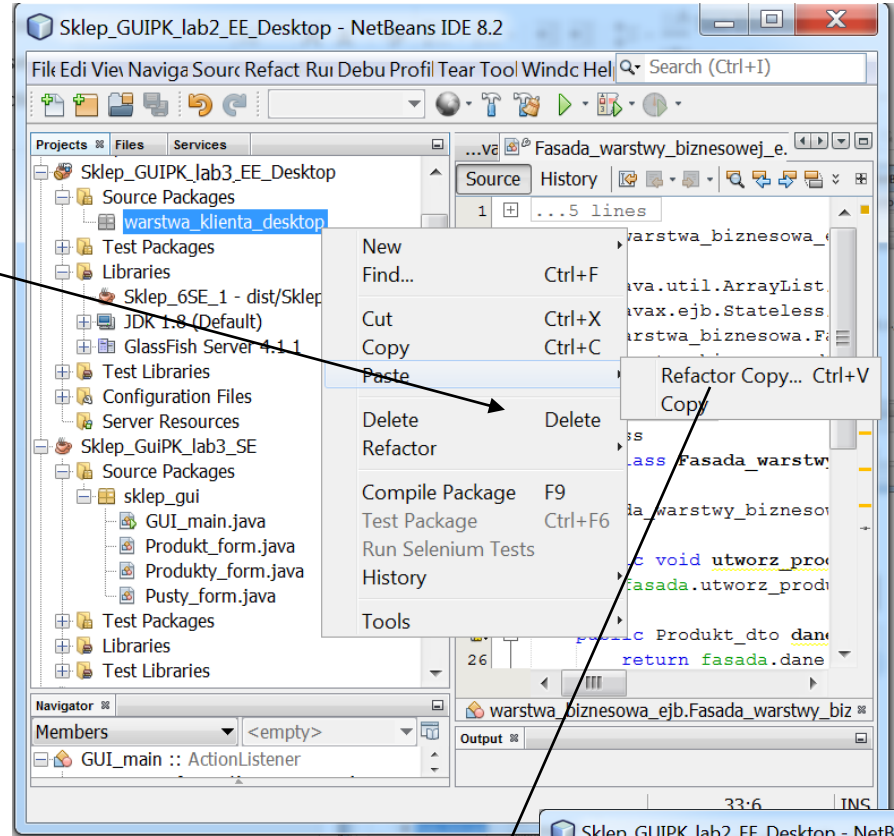
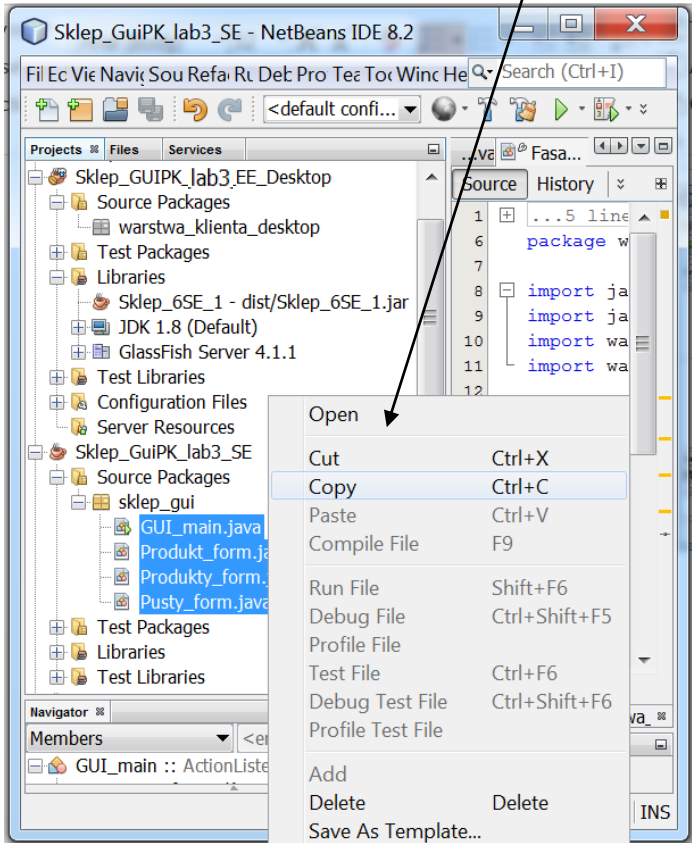
1.6. Należy wykonać projekt desktopowej warstwy klienta typu **Enterprise Application Client**:
Files/New Project/Java EE/ Enterprise Application Client o nazwie **Sklep_GUIPK_lab3_EE_Desktop** tworząc klasę **GUI_main** w pakiecie **warstwa_klienta_desktop**



1.6. cd. Należy usunąć utworzoną klasę GUI_main (pakiet **warstwa_klienta_desktop** jest pusty). Do modułu **EJB.Sklep_GUIPK_lab3_EE_Desktop** należy dodać w katalogu **Libraries** projekt **Sklep_6SE_1** (podobnie jak na str. 12) – jest to przygotowanie do skopiowania formularzy z projektu **Sklep_GuiPK_lab3_SE**



1.6. cd. Należy bezpiecznie skopiować formularze z projektu Sklep_GuiPK_lab3_SE do projektu Sklep_GUIPK_lab3_EE_Desktop (Copy/Paste/Refactor)



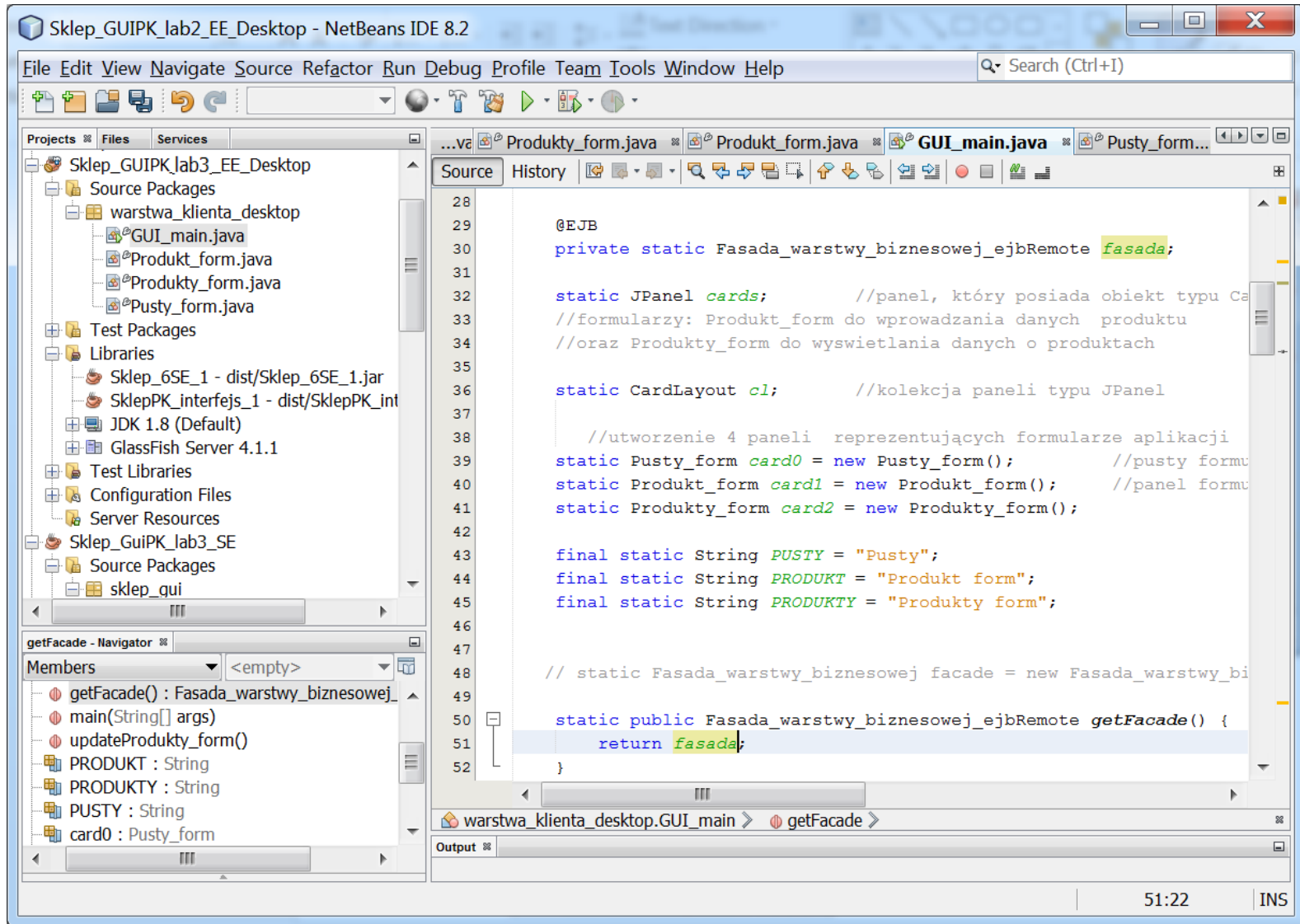
1.6. cd. W kolejnym kroku należy wstrzyknąć powiązanie w klasie GUI_main do komponentu Fasada_warstwy_biznesowej_ejb (Insert Code.../Call Enterprise Bean i wybór komponentu)

The screenshot displays the NetBeans IDE 8.2 interface. The main editor shows the source code of the `GUI_main` class, which implements `ActionListener`. The code includes static variables for `JPanel` cards, `CardLayout`, and `Pusty_form`, as well as constants for `PUSTY`, `PRODUKT`, and `PRODUKTY`. A comment indicates the creation of four panels representing form elements. The `getFacade` method is partially visible, returning `facade`.

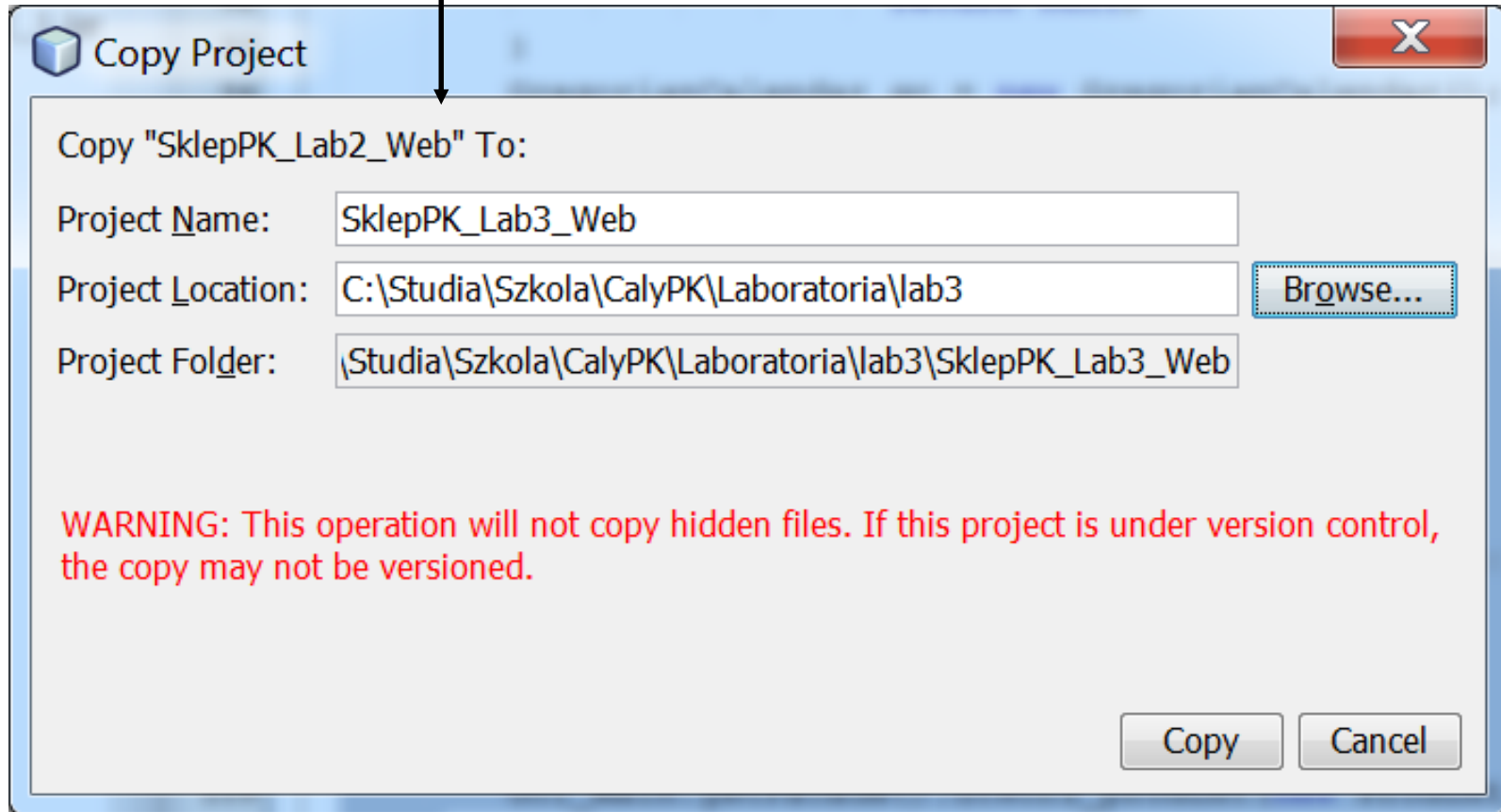
A context menu is open over the `Insert Code...` option, listing various actions such as `Navigate`, `Show Javadoc`, `Find Usages`, `Call Hierarchy`, `Insert Code...` (highlighted), `Fix Imports`, `Refactor`, `Format`, `Run File`, `Debug File`, `Test File`, `Debug Test File`, `Run Focused Test Method`, `Debug Focused Test Method`, `Run Into Method`, `New Watch...`, `Toggle Line Breakpoint`, and `Profile`.

The `Call Enterprise Bean` dialog is also open, showing a list of available beans from open projects. The selected bean is `Fasada_warstwy_biznesowej_ejb`. The `Reference Name` is set to `Fasada_warstwy_biznesowej_ejb`, and the `Referenced Interface` is set to `Remote`.

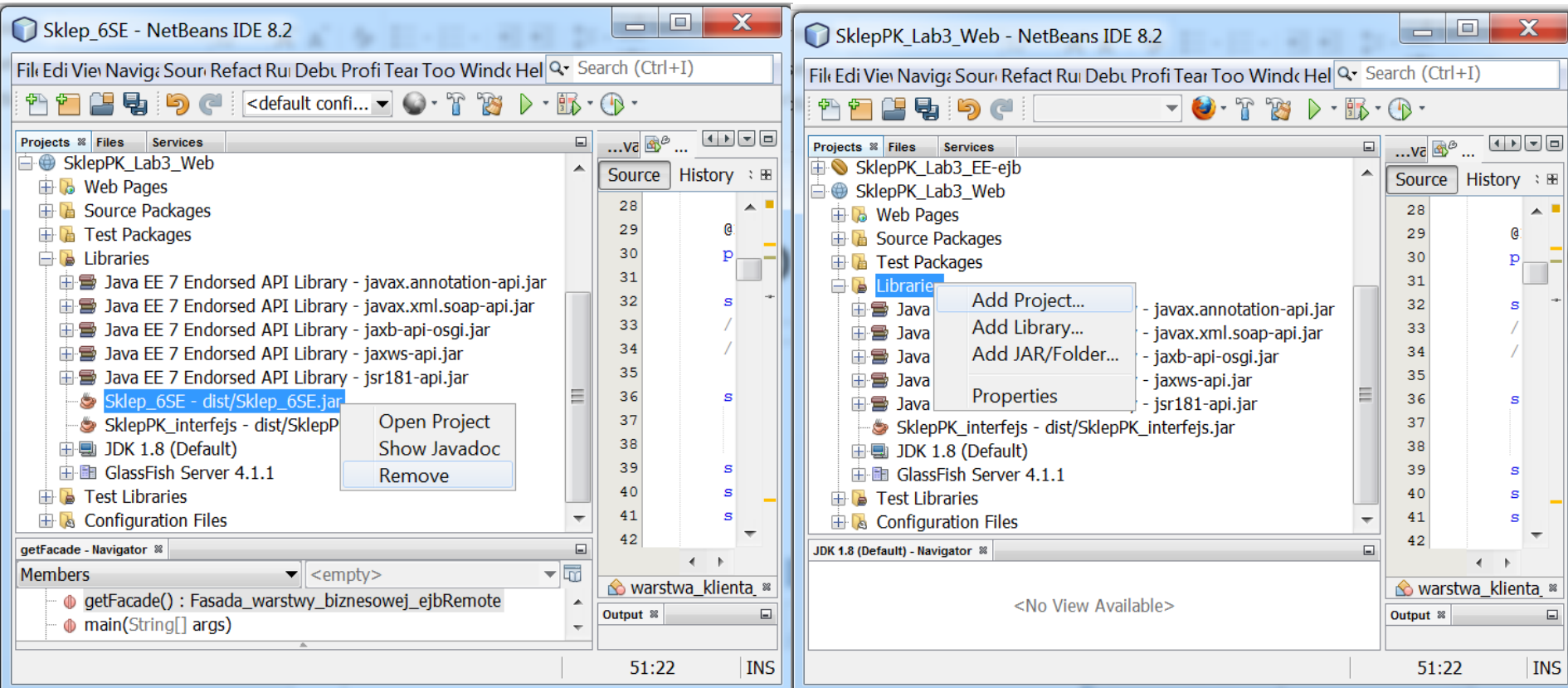
1.6. cd. Rezultat

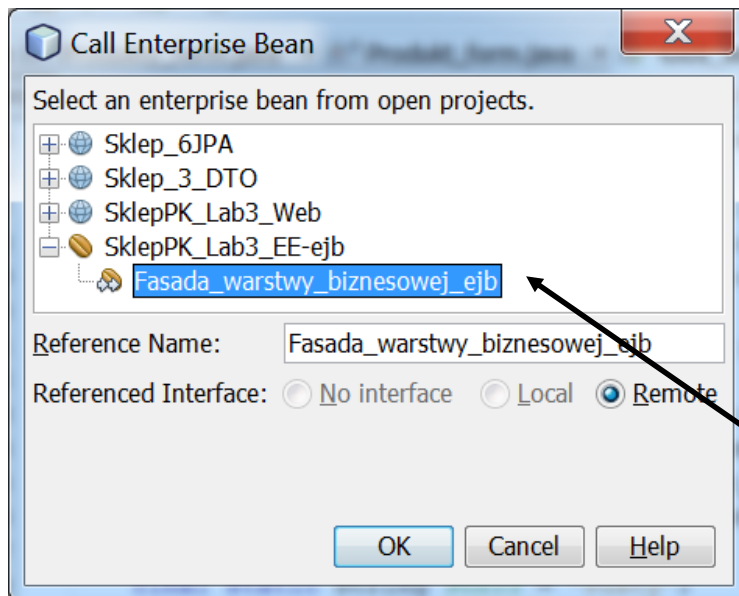
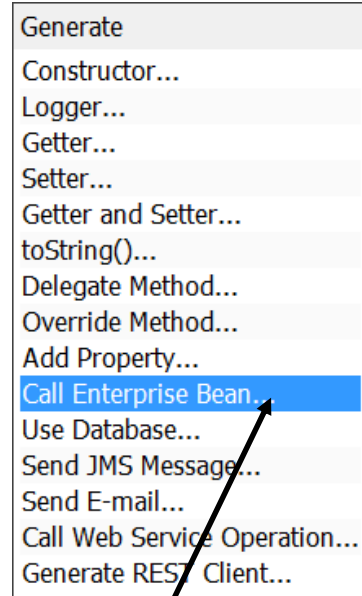
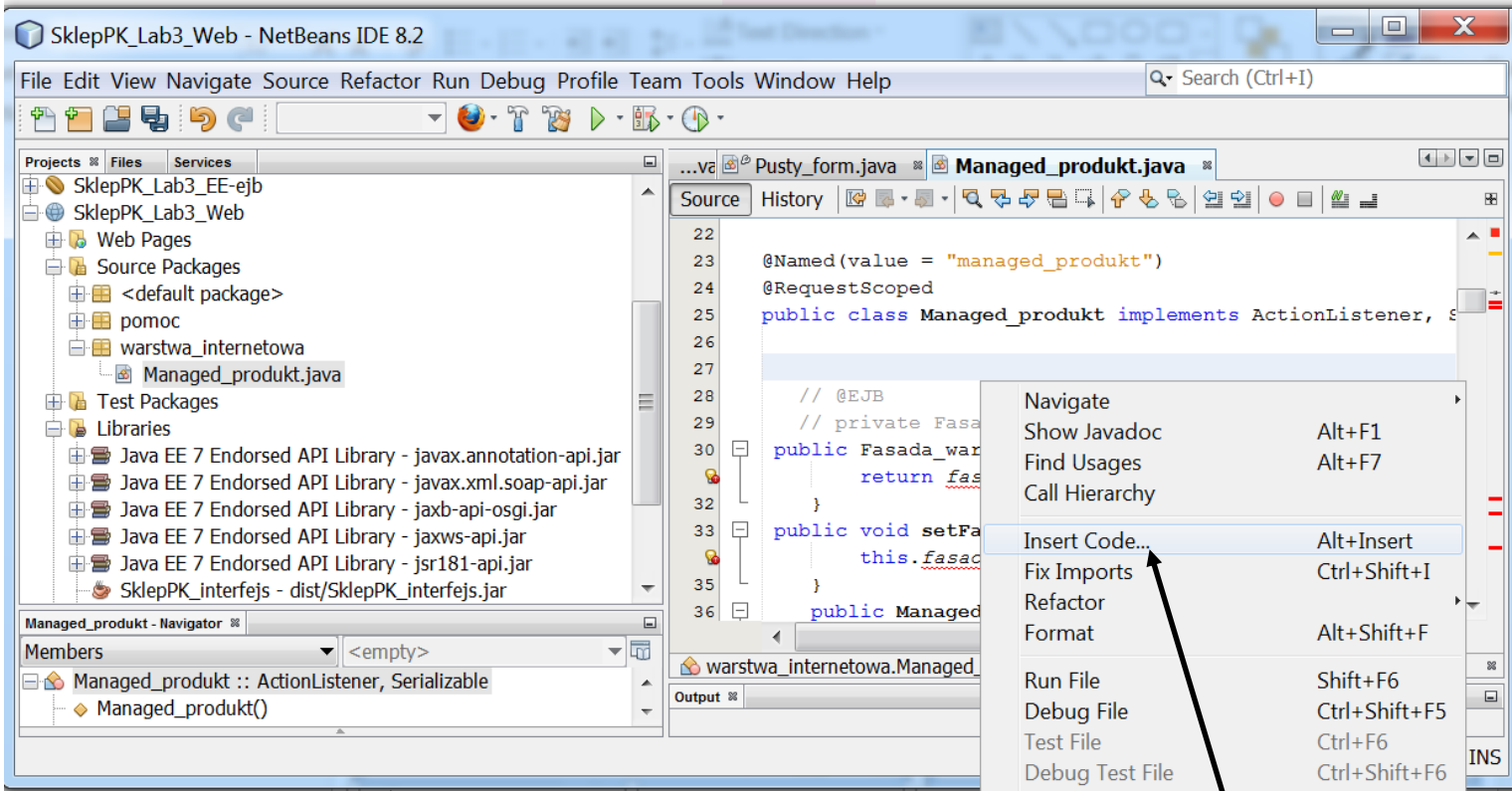


1.7. Należy wykonać kopię programu **SklepPK_Lab2_Web** jako **SklepPK_Lab3_Web** (po wykonaniu kopii zamknąć program źródłowy) – budowa odpowiada wersji projektu **Sklep_6** wykonanego wg instrukcji (str. 21) http://zofia.kruczkiewicz.staff.iar.pwr.wroc.pl/wyklady/ti /LAB_TINT_5.pdf.



1.7. cd. W projekcie **SklepPK_Lab3_Web** należy usunąć dotychczasowe powiązanie z projektem **Sklep_6SE** i zamienić je na powiązanie z **Sklep_6SE_1** w celu uzyskania dostępu do definicji klasy obiektu transferowego **Produkt_dto** (str. 12).





1.7. cd. Należy teraz „wstrzyknąć” kod nowego komponentu typu EJB z modułu **SklepPK_Lab3_EE-ejb**: kliknąć prawym klawiszem myszy na powierzchni edytora klasy **Managed_produkkt**, wybrać pozycję **Insert Code...** i wybrać z listy komponent EJB **Fasada_warstwy_biznesowej_ejb**

1.7. cd. W kolejnym kroku należy usunąć niepotrzebne powiązanie do projektu **SklepPK_Interfejs (Libraries/Remove)**. Po tych czynnościach projekt **SklepPK_Lab3_Web** jest powiązany z aktualnymi elementami nowej wielowarstwowej aplikacji EE.

The screenshot displays the NetBeans IDE 8.2 interface. The main window shows the 'SklepPK_interfejs' project with several open Java files: 'Produkt_form.java', 'GUI_main.java', 'Pusty_form.java', and 'Managed_produk...'. The 'Libraries' section in the left sidebar is selected, and a context menu is open over the 'SklepPK_interfejs - dist/SklepPK_interfejs.jar' entry, with the 'Remove' option highlighted. The 'getFasada - Navigator' window shows the members of the 'Fasada_warstwy_biznesowej_ejbRemote' interface, including methods like 'getFasada()', 'getItems()', 'getMax()', 'getMin()', 'getNazwa()', 'getNumber_convert()', 'getPromocja()', 'getStan()', 'getZmiana1()', and 'getZmiana2()'. The 'JDK 1.8 (Default) - Navigator' window shows the project structure, including 'Web Pages', 'Source Packages', 'pomoc', 'warstwa_internetowa', 'Managed_produk...java', 'Test Packages', and 'Libraries'. The 'Libraries' section is expanded, showing various Java EE 7 Endorsed API Libraries, 'Sklep_6SE_1 - dist/Sklep_6SE_1.jar', 'SklepPK_interfejs_1 - dist/SklepPK_interfejs_1.jar', and 'JDK 1.8 (Default)'.

```
package warstwa_internetowa;
```

```
import java.io.Serializable;
import java.util.Date; Należy
import javax.ejb.EJB;
import javax.enterprise.context.RequestScoped;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.faces.convert.NumberConverter;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ActionEvent;
import javax.inject.Named;
import javax.faces.event.ActionListener;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import pomoc.Zmiana_danych;
import warstwa_biznesowa_ejb.Fasada_warstwy_biznesowej_ejbRemote;
```

```
@Named(value = "managed_produkt")
```

```
@RequestScoped
```

```
public class Managed_produkt implements ActionListener, Serializable {
```

```
    @EJB
```

```
    private Fasada_warstwy_biznesowej_ejbRemote fasada;
```

```
    public Fasada_warstwy_biznesowej_ejbRemote getFasada()           {return fasada; }
```

```
    public void setFasada(Fasada_warstwy_biznesowej_ejbRemote fasada) {this.fasada = fasada; }
```

```
    public Managed_produkt() { }
```

1.7. cd. Należy zaktualizować kod obiektu typu **Managed_produkt** w projekcie **SklepPK_Lab3_Web**

```
private Produkt_dto produkt_dto = new Produkt_dto();
```

```
private DataModel items;
```

```
private int stan = 1;
```

```
private Zmiana_danych zmiana1 = new Zmiana_danych("nazwa");
```

```
private Zmiana_danych zmiana2 = new Zmiana_danych("cena");
```

```
private NumberConverter number_convert = new NumberConverter();
```

```
public Zmiana_danych getZmiana1() { return zmiana1; }
```

```
public void setZmiana1(Zmiana_danych zmiana2) { this.zmiana1 = zmiana2; }
```

```
public Zmiana_danych getZmiana2() { return zmiana2; }
```

```
public void setZmiana2(Zmiana_danych zmiana2) { this.zmiana2 = zmiana2; }
```

```
public NumberConverter getNumber_convert() {
```

```
    this.number_convert.setPattern("#####.## zł");
```

```
    return number_convert;
```

```
}
```

```
public void setNumber_convert(NumberConverter Number_convert) {
```

```
    this.number_convert = Number_convert;
```

```
}
```

```
public int getMin() { return 0; }
```

```
public int getMax() { return 100; }
```

```
public int getStan() { return stan; }
```

```
public void setStan(int stan) { this.stan = stan; }
```

```
public void setItems(DataModel items) { this.items = items; }
```

```
public DataModel utworz_DataModel() { return new ListDataModel(fasada.items()); }
```

```
public DataModel getItems() {
```

```
    if (items == null)
```

```
        items = utworz_DataModel();
```

```
    return items;
```

```
}
```

1.7. cd. W komponencie **Managed_produk** należy zastąpić model komponentów widoków obiektem transferowym typu **Produkt_dto** tak, aby nie zmieniać kodu plików znacznikowych: **dodaj_produk1.xhtml**, **rezultat2.xhtml**. Ulegnie zmianie kod pliku **lista_produkow.xhtml**

1.7. cd. Aktualizacja kodu – nagłówki metod dostępu do modeli komponentów JSF nie uległy zmianie

```
public String getNazwa()
public void setNazwa(String nazwa)
public float getCena()
public void setCena(float cena)
public int getPromocja()
public void setPromocja(int promocja)
public float getCena_brutto()
public void setCena_brutto(float cena_brutto)
public Date getData_produkcji()
public void setData_produkcji(Date data_produkcji)
```

```
{ return produkt_dto.getNazwa(); }
{ this.produkt_dto.setNazwa(nazwa); }
{ return produkt_dto.getCena(); }
{ this.produkt_dto.setCena(cena); }
{ return produkt_dto.getPromocja(); }
{ this.produkt_dto.setPromocja(promocja); }
{ return produkt_dto.getCena_brutto(); }
{ this.produkt_dto.setCena_brutto(cena_brutto); }
{ return produkt_dto.getData_produkcji(); }
{ this.produkt_dto.setData_produkcji(data_produkcji); }
```

```
public void dodaj_produkt() {
    fasada.utworz_produkt(produkt_dto);
    dane_produktu();
}
public void dane_produktu() {
    stan = 1;
    produkt_dto = fasada.dane_produktu();
    if (produkt_dto == null)
        stan = 0;
}
```

Metody obsługujące wysyłanie i pobieranie danych z fasady sesyjnej logiki biznesowej oparte są teraz na obiekcie transferowym Produkt_dto.

1.7. cd. Ten kod w klasie Managed_produkci nie uległ zmianie – podany jest dla przypomnienia.

@Override

```
public void processAction(ActionEvent event) throws AbortProcessingException {  
    dodaj_produkci();  
}
```

```
public void zakrespromocji(FacesContext context,   UIComponent toValidate, Object value)  
{  
    stan = 1;  
    int input = ((Long) value).intValue();  
    if (input < getMin() || input > getMax()) {  
        ((UIInput) toValidate).setValid(false);  
        FacesMessage message = new FacesMessage("Dane poza zakresem");  
        context.addMessage(toValidate.getClientId(context), message);  
        stan = 0;  
    }  
}  
}
```

1.7. cd. Modyfikacja jedynie pliku **lista_produkтов.xhtml** w celu zastawania kolekcji obiektów transferowych typu **Produkt_dto**, przekazanej za pomocą metody **getItems**, jako modelu danych komponentu typu **dataTable**

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:h="http://xmlns.jcp.org/jsf/html"
xmlns:f="http://xmlns.jcp.org/jsf/core">
<body>
<ui:composition template=" ../template.xhtml">
<ui:define name="title">
<h:outputText value="#{bundle['lista_produkтов.tytul']}"></h:outputText>
</ui:define>
<ui:define name="content">
<h:form styleClass="jsfcrud_list_form">
<h:outputText escape="false" value="#{bundle['lista_produkтов.pusta']}"
rendered="#{managed_produkтов.items.rowCount == 0}"/>
<h:panelGroup rendered="#{managed_produkтов.items.rowCount > 0}">
<h:dataTable value="#{managed_produkтов.items}" var="item" border="0"
cellpadding="2" cellspacing="0"
rowClasses="jsfcrud_odd_row,jsfcrud_even_row"
rules="all" style="border:solid 1px">
```

1.7. cd. Modyfikacja jedynie pliku **lista_produkow.xhtml** w celu zastawiania kolekcji obiektów transferowych typu **Produkt_dto**, przekazanej za pomocą metody **getItems**, jako modelu danych komponentu typu **dataTable**

```
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.id']}" /> </f:facet>
  <h:outputText value="#{item.id}" />
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.nazwa']}" /> </f:facet>
  <h:outputText value="#{item.nazwa}" />
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.cena']}" /> </f:facet>
  <h:outputText value="#{item.cena}" />
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.promocja']}" /> </f:facet>
  <h:outputText value="#{item.promocja}" />
</h:column>
<h:column>
```

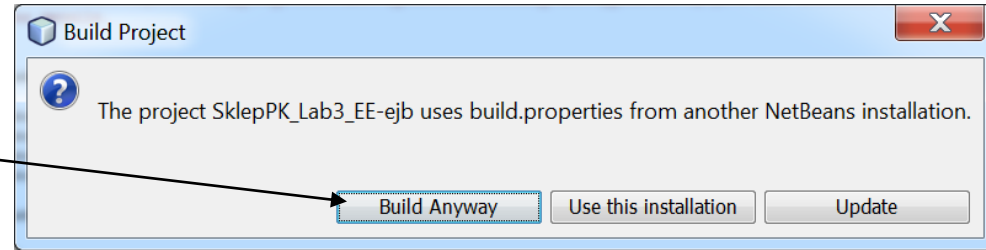
1.7. cd. Modyfikacja jedynie pliku **lista_produkow.xhtml** w celu zastawiania kolekcji obiektów transferowych typu **Produkt_dto**, przekazanej za pomocą metody **getItems**, jako modelu danych komponentu typu **dataTable**

```
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.data']}/> </f:facet>
  <h:outputText value="#{item.data_produkcji}"/>
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.cena_brutto']}/> </f:facet>
  <h:outputText value="#{item.cena_brutto}"/>
</h:column>
</h:dataTable>
</h:panelGroup>
<h:commandButton id="powrot" value="#{bundle['lista_produkow.powrot']}" action="/faces/index1" />
</h:form>
</ui:define>
</ui:composition>
</body>
</html>
```

1.8. Uruchomienie projektu.

Należy w podanej kolejności wykonać operacje **Clean and Build** na projektach składowych (w celu łatwiejszej lokalizacji błędów):

- 1) Sklep_6SE_1
- 2) SklepPK_interfejs_1
- 3) SklepPK_Lab3_EE-ejb
- 4) SklepPK_Lab3_Web
- 5) Sklep_GUIPK_lab3_EE_Desktop
- 6) SklepPK_Lab3_EE



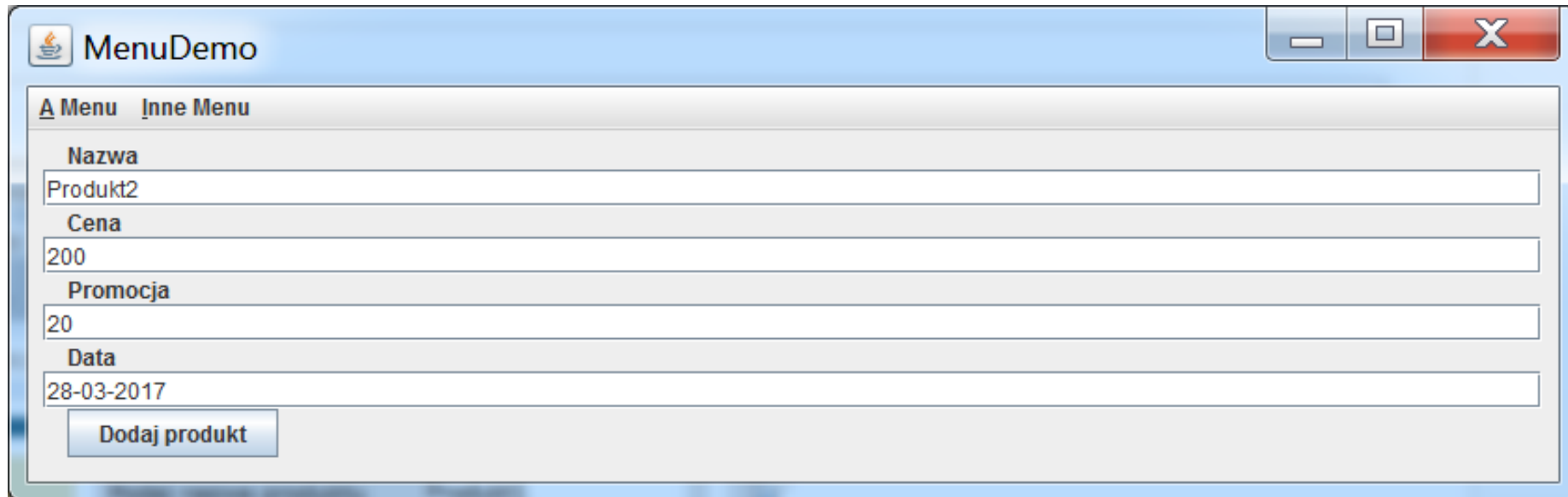
Następnie, należy wykonać operację **Deploy** na projekcie **SklepPK_Lab3_EE**.

Teraz można uruchomić dowolną liczbę aplikacji klienckich za pomocą operacji **Run**:

- 1) SklepPK_Lab3_Web (**pozostałe instancje w kolejnych instancjach przeglądark: [http://localhost:8080/ SklepPK_Lab3_Web/](http://localhost:8080/SklepPK_Lab3_Web/)**)
- 2) Sklep_GUIPK_lab3_EE_Desktop

W przykładzie uruchomiono jedną instancję aplikacji desktopowej i dwie instancje aplikacji internetowej.

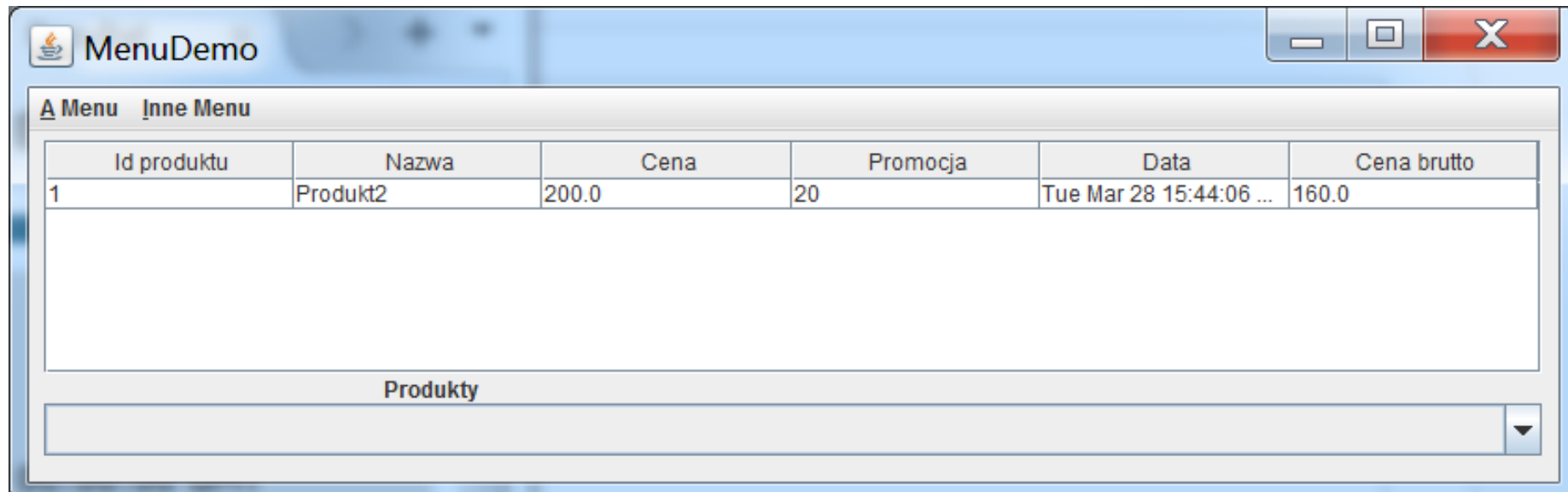
1.8. cd. Uruchomienie aplikacji EE typu **SklepPK_Lab3_EE** z modułem **SklepPK_Lab3_EE-ejb** udostępniającym logikę biznesową projektu **Sklep_6SE_1** oraz z dwoma typami klientów: internetowym **SklepPK_Lab3_Web** i desktopowym **Sklep_GUIPK_lab3_EE_Desktop** – wprowadzanie danych



The screenshot shows a desktop application window titled "MenuDemo". The window contains a form with the following fields and values:

- A Menu** | **Inne Menu**
- Nazwa**: Produkt2
- Cena**: 200
- Promocja**: 20
- Data**: 28-03-2017

At the bottom of the form is a button labeled "Dodaj produkt".

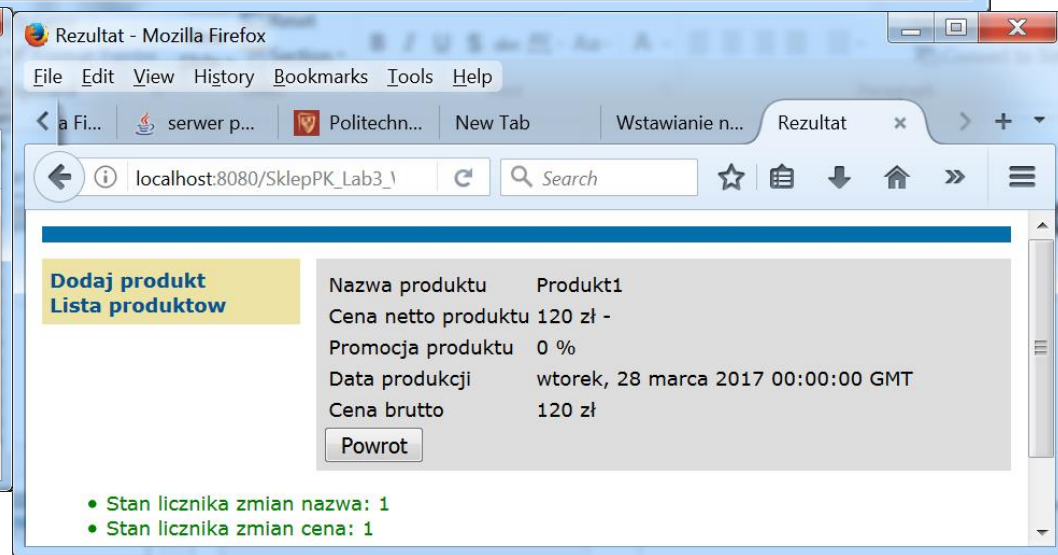
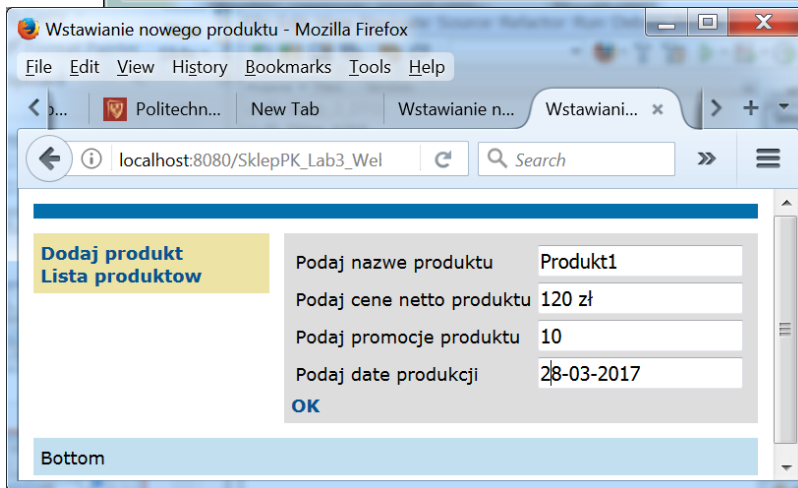
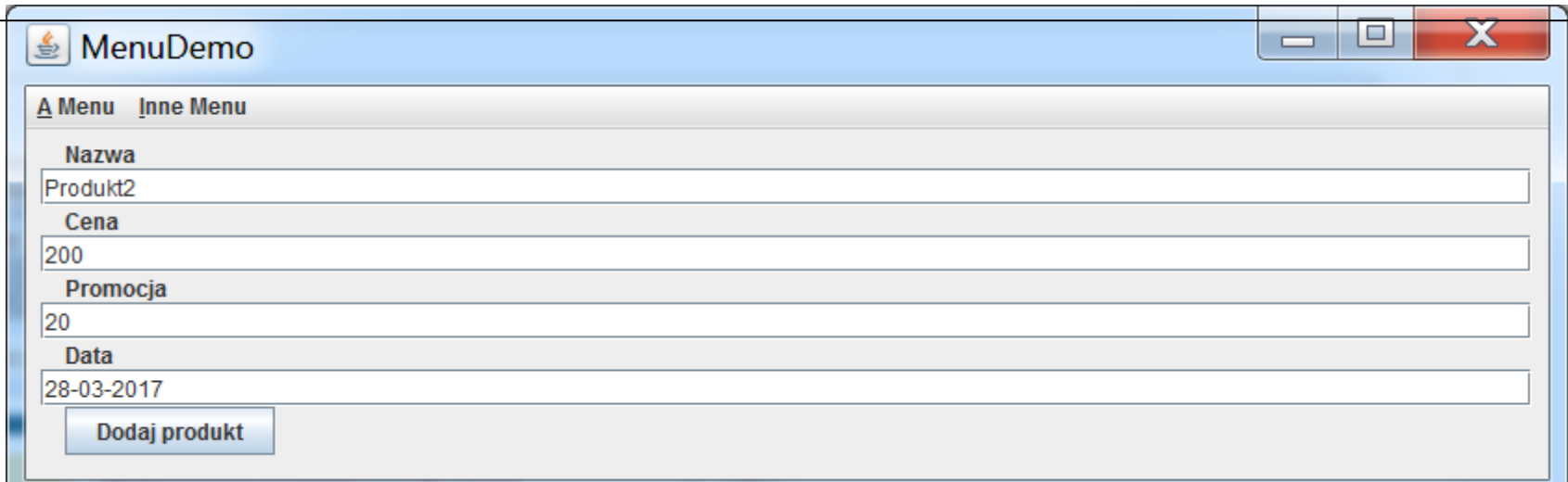


The screenshot shows the same "MenuDemo" application window, but now displaying a table of products. The table has the following columns and data:

Id produktu	Nazwa	Cena	Promocja	Data	Cena brutto
1	Produkt2	200.0	20	Tue Mar 28 15:44:06 ...	160.0

Below the table is a label "Produkty" and a scroll bar.

1.8. cd. Uruchomienie aplikacji EE typu **SklepPK_Lab3_EE** z modułem **SklepPK_Lab3_EE-ejb** udostępniającym logikę biznesową projektu **Sklep_6SE_1** oraz z dwoma typami klientów: internetowym **SklepPK_Lab3_Web** i desktopowym **Sklep_GUIPK_lab3_EE_Desktop**



1.8. cd. Uruchomienie aplikacji EE typu **SklepPK_Lab3_EE** z modułem **SklepPK_Lab3_EE-ejb** udostępniającym logikę biznesową projektu **Sklep_6SE_1** oraz z dwoma typami klientów: internetowym **SklepPK_Lab3_Web** i desktopowym **Sklep_GUIPK_lab3_EE_Desktop**

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow x Lista produktow x +

localhost:8080/SklepPK_Lab3_Web/faces/war Search

Dodaj produkt
Lista produktow

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt2	200.0	20	Tue Mar 28 17:08:42 CEST 2017	160.0
2	Produkt1	120.0	10	Tue Mar 28 02:00:00 CEST 2017	108.0

Powrot

Bottom

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow x Lista produktow x +

localhost:8080/SklepPK_Lab3_Web/faces/war Search

Dodaj produkt
Lista produktow

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt2	200.0	20	Tue Mar 28 17:08:42 CEST 2017	160.0
2	Produkt1	120.0	10	Tue Mar 28 02:00:00 CEST 2017	108.0

Powrot

Bottom

MenuDemo

A Menu Inne Menu

Id produktu	Nazwa	Cena	Promocja	Data	Cena brutto
1	Produkt2	200.0	20	Tue Mar 28 17:08:42 CEST 2017	160.0
2	Produkt1	120.0	10	Tue Mar 28 02:00:00 CEST 2017	108.0

Produkty

[1, Produkt2, 200.0, 20, Tue Mar 28 17:08:42 CEST 2017, 160.0]

[1, Produkt2, 200.0, 20, Tue Mar 28 17:08:42 CEST 2017, 160.0]

[2, Produkt1, 120.0, 10, Tue Mar 28 02:00:00 CEST 2017, 108.0]

2. Dodanie stronicowania zawartości komponentu **dataTable** na stronie **lista_produkow.xhtml**.

Kod dotyczący stronicowania oparty na kodzie wygenerowanym podczas tworzenia stron JSF na podstawie klas typu Entity (pakiet jsf.util) –

Dodatek do wykładu 2 (str. 32-85):

http://zofia.kruckiewicz.staff.iiar.pwr.wroc.pl/wyklady/ti_/TINT_2.pdf

2.1. Dodanie stronicowania stron – fragment pliku [lista_produkow.xhtml](#) z dodanym kodem znaczników odpowiedzialnym za stronicowanie (zaznaczony kolorem czerwonym – przed znacznikiem dataTable) – pierwszy znacznik `commandLink` służy do stronicowania „wstecz” a drugi do stronicowania „do przodu”

```
<ui:define name="content">
  <h:form styleClass="jsfcrud_list_form">
    <h:panelGroup id="messagePanel" layout="block">
      <h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
    </h:panelGroup>
    <h:outputText escape="false" value="#{bundle['jsf.lista_produkow.pusta']}"
      rendered="#{managed_produkow.pagination.itemsCount == 0}"/>
    <h:panelGroup rendered="#{managed_produkow.pagination.itemsCount > 0}">
      <h:outputText value="#{managed_produkow.pagination.pageFirstItem + 1}
        ..#{managed_produkow.pagination.pageLastItem + 1}
        /#{managed_produkow.pagination.itemsCount}"/>&nbsp;
      <h:commandLink
        action="#{managed_produkow.previous}"
        value="#{bundle['jsf.lista_produkow.poprzedni']} #{managed_produkow.pagination.pageSize}"
        rendered="#{managed_produkow.pagination.hasPreviousPage}"/>&nbsp;
      <h:commandLink
        action="#{managed_produkow.next}"
        value="#{bundle['jsf.lista_produkow.nastepny']} #{managed_produkow.pagination.pageSize}"
        rendered="#{managed_produkow.pagination.hasNextPage}"/>&nbsp;
    </h:panelGroup>
  </h:form>
</ui:define>
```

2.1. cd. Zmiana czasu życia obiektu typu **Managed_produk** do czasu trwania sesji za pomocą adnotacji **@SessionScoped**

```
package warstwa_internetowa;
```

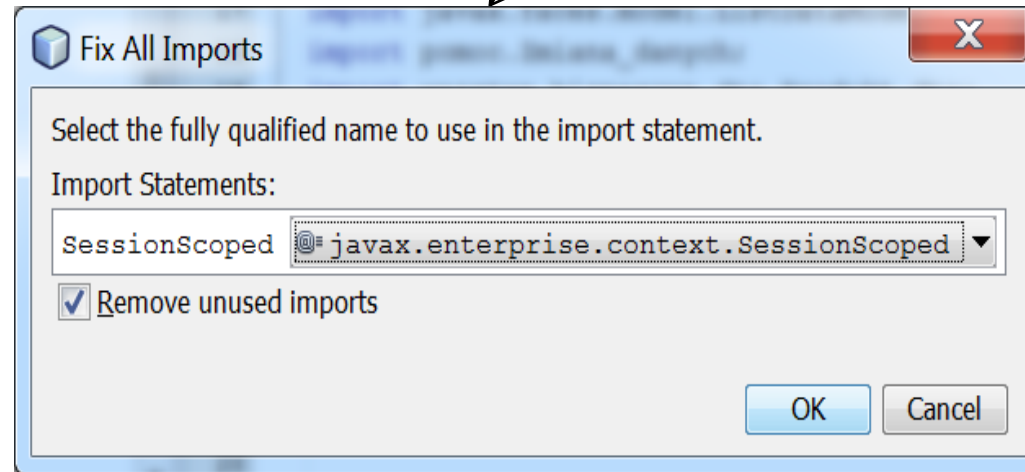
```
import java.io.Serializable;  
import java.util.Date;  
import javax.ejb.EJB;  
import javax.enterprise.context.SessionScoped;  
import javax.faces.application.FacesMessage;  
import javax.faces.component.UIComponent;  
import javax.faces.component.UIInput;  
import javax.faces.context.FacesContext;  
import javax.faces.convert.NumberConverter;  
import javax.faces.event.AbortProcessingException;  
import javax.faces.event.ActionEvent;  
import javax.inject.Named;  
import javax.faces.event.ActionListener;  
import javax.faces.model.DataModel;  
import javax.faces.model.ListDataModel;  
import pomoc.Zmiana_danych;  
import warstwa_biznesowa.dto.Produkt_dto;  
import warstwa_biznesowa_ejb.Fasada_warstwy_biznesowej_ejbRemote;
```

```
@Named(value = "managed_produk")
```

```
@SessionScoped
```

```
public class Managed_produk implements ActionListener, Serializable {
```

Podczas wprowadzania importu brakujących klas za pomocą **Fix Import** (wybrana pozycja po kliknięciu prawym klawiszem na powierzchnię okna edytora)



2.1. cd. Rezultat zmian

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

```
1 package warstwa_internetowa;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.ejb.EJB;
6 import javax.enterprise.context.SessionScoped;
7 import javax.faces.application.FacesMessage;
8 import javax.faces.component.UIComponent;
9 import javax.faces.component.UIInput;
10 import javax.faces.context.FacesContext;
11 import javax.faces.convert.NumberConverter;
12 import javax.faces.event.AbortProcessingException;
13 import javax.faces.event.ActionEvent;
14 import javax.inject.Named;
15 import javax.faces.event.ActionListener;
16 import javax.faces.model.DataModel;
17 import javax.faces.model.ListDataModel;
18 import pomoc.Zmiana_danych;
19 import warstwa_biznesowa.dto.Produkt_dto;
20 import warstwa_biznesowa_ejb.Fasada_warstwy_biznesowej_ejbRemote;
21
22 @Named(value = "managed_produkkt")
23 @SessionScoped
24 public class Managed_produkkt implements ActionListener, Serializable {
```

The IDE interface includes a Project Explorer on the left showing the project structure, a Navigator at the bottom left, and an Output window at the bottom right. The Output window shows the following processes: Java DB Database Process, GlassFish Server 4.1.1, Sklep_GUIPK_lab3_EE_Desktop (run), and Sklep...

2.1. cd. Kod, który należy dodać i zmodyfikować w klasie **Managed_produk**t.

```
public DataModel getItems() {  
    if (items == null || fasada.isStan()) {  
        items = getPagination().createPageDataModel(); }  
    return items;  
}  
  
public void dodaj_produk(t) {  
    fasada.utworz_produk(t, produkt_dto);  
    dane_produk(t);  
    recreateModel();  
    getPagination().nextPage();  
}  
  
public void dane_produk(t) {  
    stan = 1;  
    produkt_dto = fasada.dane_produk(t);  
    if (produkt_dto == null) {  
        produkt_dto = new Produkt_dto();  
        stan = 0; }  
}
```

W przypadku, gdy komponent typu `Managed_produk(t)` ma czas życia sesji, należy w sposób algorytmiczny aktualizować dane produktów za pomocą metody `getItems` do wyświetlania, pobierając z miejsca przechowywania, czyli z serwera, gdy:

- badanie `items==null` wynika z konieczności pobrania danych do widoku komponentu `dataTable`, gdy dodano nowe dane produktów z klienckiej aplikacji internetowej () – metoda `recreateModel` ustawia wartość `items` na `null`,
- a warunek `fasada.isStan()` wynika z badania, czy z aplikacji desktopowej nie wstawiono nowych danych.

```
private PaginationHelper pagination;
```

```
//atrybut klasy Managed_produkt
```

```
public PaginationHelper getPagination() {
```

```
    if (pagination == null) {
```

```
        pagination = new PaginationHelper(3) {
```

```
            @Override
```

```
            public int getItemsCount() {
```

```
                return getFasada().count();
```

```
            }
```

```
            @Override
```

```
            public DataModel createPageDataModel() {
```

```
                int[] range = {getPageFirstItem(), getPageLastItem() + 1};
```

```
                return new ListDataModel(getFasada().findRange(range));
```

```
            }
```

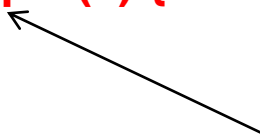
```
        };
```

```
    }
```

```
    return pagination;
```

```
}
```

Ustalenie wartości atrybutu **pageSize**, w obiekcie klasy dziedziczącej po klasie abstrakcyjnej **PaginationHelper**, obsługującej stronicowanie komponentu **dataTable** (definicję klasy podano na kolejnych slajdach) z zaimplementowanymi metodami **getItemsCount** oraz **createPageDataModel**



```
private void recreateModel() {  
    items = null;  
}
```

Metoda usuwająca zawartość bieżącej strony – umożliwia metodzie **getItems** utworzenie zawartości wybranej strony w sposób algorytmiczny

```
public String next() {  
    getPagination().nextPage();  
    recreateModel();  
    return "lista_produktow";  
}
```

Metoda obsługująca wybór następnej strony

```
public String previous() {  
    getPagination().previousPage();  
    recreateModel();  
    return "lista_produktow";  
}
```

Metoda obsługująca wybór poprzedniej strony

2.1. cd. Do pakietu **pomoc** należy dodać klasę **PaginationHelper**, typu **abstract** zawierającą następujący kod:

```
package pomoc;  
import javax.faces.application.FacesMessage;  
import javax.faces.context.FacesContext;
```

```
public abstract class PaginationHelper {
```

```
    private int pageSize;
```

```
    private int page;
```

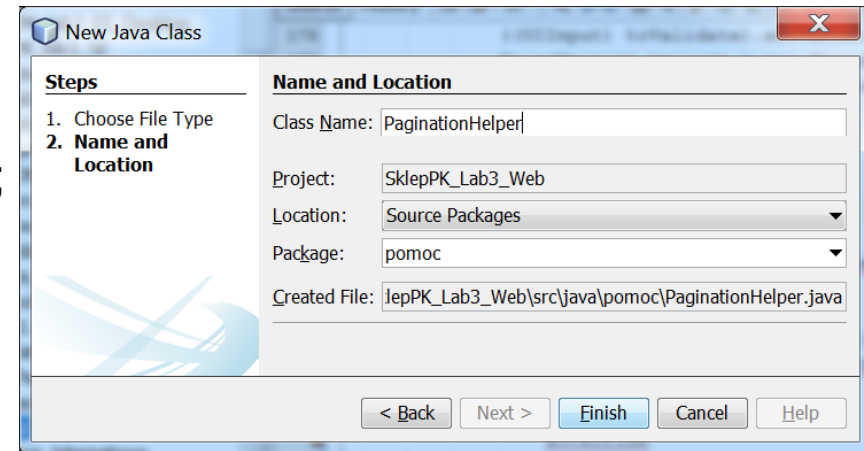
```
    public PaginationHelper(int pageSize) {  
        this.pageSize = pageSize;
```

```
    }
```

```
    public abstract int getItemCount();
```

```
    public abstract DataModel createPageDataModel();
```

```
    public int getPageFirstItem() {  
        return page * pageSize; }  
}
```



pageSize - rozmiar strony
– czyli liczba wierszy
komponentu **dataTable**
page – numer strony

pageSize*page – liczba
pozycji przypisanych do
wszystkich stron, a
jednocześnie numer
danej wyświetlanej jako
pierwsza pozycja na
ostatniej stronie

```
public int getPageLastItem() {  
    int i = getPageFirstItem() + pageSize - 1;  
    int count = getItemsCount() - 1;  
    if (i > count) {  
        i = count; }  
    if (i < 0) {i = 0; }  
    return i;  
}
```

Indeks ostatniej pozycji, czyli numer ostatniej danej, wyświetlany na ostatniej stronie

```
public boolean hasNextPage() {  
    return (page + 1) * pageSize + 1 <= getItemsCount();  
}
```

Sprawdzenie, czyli liczba danych wymaga utworzenia następnej strony

```
public void nextPage() {  
    if (hasNextPage()) {  
        page++;  
    }  
}
```

Numer ostatniej strony


```
public boolean isHasPreviousPage() {  
    return page > 0;  
}
```

Sprawdzenie, czy istnieje poprzednia strona (pierwsza strona ma numer 0)

```
public void previousPage() {  
    if (isHasPreviousPage()) {  
        page--;  
    }  
}
```

Pobranie numeru poprzedniej strony

```
public int getPageSize() {  
    return pageSize;  
}
```

Pobranie rozmiaru strony

```
public void setPage() {  
    this.page = getItemCount()/pageSize;  
}
```

Aktualizacja liczby stron

2.2. Kod, który należy dodać do klasy **Fasada_warstwy_biznesowej**

```
public boolean isStan() {  
    return stan;  
}  
public void setStan(boolean stan) {  
    this.stan = stan;  
}  
public int count() {  
    return produkty.size();  
}  
public ArrayList<Produkt_dto> findRange(int[] range) {  
    ArrayList<Produkt_dto> pom = new ArrayList();  
    for (int i = range[0]; i < range[1]; i++) {  
        pom.add(produkt_transfer(getProdukty().get(i)));  
    }  
    return pom;  
}
```

Pobranie informacji, czy dodano nowe obiekty typu **Produkt1** do kolekcji produkty

Pobranie podzbioru danych potrzebnych do wyświetlenia na stronie za pomocą metody **findRange**. Tablica **range** zawiera dwa elementy: pierwszy zawiera numer pierwszego elementu, drugi element zawiera numer ostatniego elementu z kolekcji **produkty**, które wyznaczają podzbiór danych pobieranych do wyświetlenia na stronie. Pobrane elementy z kolekcji produktu są przekształcone na obiekty transferowe typu **Produkt_dto**.

2.3. Kod, który należy dodać do interfejsu **Fasada_warstwy_biznesowej_ejbRemote**

@Remote

```
public interface Fasada_warstwy_biznesowej_ejbRemote {  
    public void utworz_produkt(Produkt_dto produkt_dto);  
    public Produkt_dto dane_produktu();  
    public ArrayList<ArrayList<String>> items();  
    public ArrayList<Produkt_dto> items_();  
    public int count();  
    public ArrayList<Produkt_dto> findRange(int[] range);  
    public boolean isStan() ;  
    public void setStan(boolean stan);  
}
```

2.4. Kod, który należy dodać do klasy **Fasada_warstwy_biznesowej_ejb**, pełniącej rolę tzw sesyjnej fasady logiki biznesowej, realizowanej przez obiekt typu **Fasada_warstwy_biznesowej**

```
public ArrayList<Produkt_dto> findRange(int[] range) {  
    return fasada.findRange(range);  
}
```

```
public int count() {  
    return fasada.count();  
}
```

```
public void setStan(boolean stan) {  
    fasada.setStan(stan);  
}
```

```
public boolean isStan() {  
    return fasada.isStan();  
}
```

Pośredniczenie w przekazaniu podzbioru danych z logiki biznesowej do aplikacji klienckich: desktopowej i internetowej

2.5. Uzupełnienie zawartości pliku **Bundle.properties**

jsf.lista_produkow.poprzedni=Poprzedni

jsf.lista_produkow.nastepny=Nastepny

2.5. cd. Wstawienie znaczników do konwersji danych w pliku **lista_produkow.xhtml** – dzięki temu, że wyświetlane są atrybuty obiektu typu **Produkt_dto** o różnych typach (int, float, Date), przekazane za pomocą metod typu **getter** obiektu typu **Produkt_dto**.

```
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.id']}" /> </f:facet>
  <h:outputText value="#{item.id}" />
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.nazwa']}" /> </f:facet>
  <h:outputText value="#{item.nazwa}" />
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.cena']}" /> </f:facet>
  <h:outputText value="#{item.cena}" />
  <f:convertNumber pattern="###.## z&#322;" />
  </h:outputText>
</h:column>
<h:column><f:facet name="header"><h:outputText value="#{bundle['lista_produkow.promocja']}" />
</f:facet>
  <h:outputText value="#{item.promocja}" />
  <f:convertNumber currencySymbol="%" type="currency" />
  </h:outputText>
</h:column>
<h:column>
```

2.5. cd. Wstawienie znaczników do konwersji danych w pliku **lista_produkow.xhtml** – dzięki temu, że wyświetlane są atrybuty obiektu typu **Produkt_dto** o różnych typach (int, float, Date), przekazane za pomocą metod typu **getter** obiektu typu **Produkt_dto**

```
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.data']}/> </f:facet>
  <h:outputText value="#{item.data_produkcji}">
    <f:convertDateTime pattern="EEEEEEEE, dd-MM-yyyy" />
  </h:outputText>
</h:column>
<h:column>
  <f:facet name="header"> <h:outputText value="#{bundle['lista_produkow.cena_brutto']}/> </f:facet>
  <h:outputText value="#{item.cena_brutto}">
    <f:convertNumber currencySymbol="z&#322;" type="currency"/>
  </h:outputText>
</h:column>
</h:dataTable>
</h:panelGroup>
<h:commandButton id="powrot" value="#{bundle['lista_produkow.powrot']}" action="/faces/index1" />
</h:form>
</ui:define>
</ui:composition>
</body>
</html>
```


2.6. Prezentacja stronicowania (1) – należy kolejno wstawić dane nowych produktów za pomocą aplikacji klienckiej desktopowej i internetowej i wyświetlać wynik za pomocą formularzu **lista_produktyw.xhtml** w klienckiej aplikacji internetowej. Uruchomienie wg p.1.8 (str .41)

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/li

Search

Dodaj produkt
Lista produktow

1 ..1 /1

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	120 zł	10 %	wtorek, 28-03-2017	108 zł

Powrot

Bottom

Dodaj produkt
Lista produktow

1 ..2 /2

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	120 zł	10 %	wtorek, 28-03-2017	108 zł
2	Produkt2	220 zł	20 %	wtorek, 28-03-2017	176 zł

Powrot

Bottom

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/li

Search

Dodaj produkt
Lista produktow

1 ..3 /3

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	120 zł	10 %	wtorek, 28-03-2017	108 zł
2	Produkt2	220 zł	20 %	wtorek, 28-03-2017	176 zł
3	Produkt3	320 zł	25 %	wtorek, 28-03-2017	240 zł

Powrot

2.6. cd. Prezentacja stronicowania (2)

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/list

Dodaj produkt
Lista produktow

1 ..3 / 4 **Następny 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	120 zł	10 %	wtorek, 28-03-2017	108 zł
2	Produkt2	220 zł	20 %	wtorek, 28-03-2017	176 zł
3	Produkt3	320 zł	25 %	wtorek, 28-03-2017	240 zł

Powrot

Bottom

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/lis

Dodaj produkt
Lista produktow

4 ..5 / 5 **Poprzedni 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
4	Produkt4	440 zł	50 %	wtorek, 28-03-2017	220 zł
5	Produkt5	550 zł	10 %	wtorek, 28-03-2017	495 zł

Powrot

Bottom

2.6. cd. Prezentacja stronicowania (3)

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/lis

Dodaj produkt
Lista produktow

4 ..6 /6 **Popzedni 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
4	Produkt4	440 zł	50 %	wtorek, 28-03-2017	220 zł
5	Produkt5	550 zł	10 %	wtorek, 28-03-2017	495 zł
6	Produkt6	660 zł	10 %	wtorek, 28-03-2017	594 zł

Powrot

Bottom

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/lis

Dodaj produkt
Lista produktow

7 ..7 /7 **Popzedni 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
7	Produkt7	770 zł	10 %	wtorek, 28-03-2017	693 zł

Powrot

Bottom

2.6. cd. Prezentacja stronicowania (4)

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/lis

Dodaj produkt
Lista produktow

4 ..6 /7 **Popzedni 3** **Nastepny 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
4	Produkt4	440 zł	50 %	wtorek, 28-03-2017	220 zł
5	Produkt5	550 zł	10 %	wtorek, 28-03-2017	495 zł
6	Produkt6	660 zł	10 %	wtorek, 28-03-2017	594 zł

Powrot

Bottom

Lista produktow - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Lista produktow

localhost:8080/SklepPK_Lab3_Web/faces/warstwa_internetowa_jsf/lis

Dodaj produkt
Lista produktow

1 ..3 /7 **Nastepny 3**

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	120 zł	10 %	wtorek, 28-03-2017	108 zł
2	Produkt2	220 zł	20 %	wtorek, 28-03-2017	176 zł
3	Produkt3	320 zł	25 %	wtorek, 28-03-2017	240 zł

Powrot

Bottom

2.6. cd. Widok listy wprowadzonych produktów w aplikacji desktopowej

MenuDemo [Standard Windows Title Bar with Minimize, Maximize, Close buttons]

A Menu Inne Menu

Id produktu	Nazwa	Cena	Promocja	Data	Cena brutto
1	Produkt1	120.0	10	Tue Mar 28 02:00:00 CE...	108.0
2	Produkt2	220.0	20	Tue Mar 28 02:00:00 CE...	176.0
3	Produkt3	320.0	25	Tue Mar 28 02:00:00 CE...	240.0
4	Produkt4	440.0	50	Tue Mar 28 02:00:00 CE...	220.0
5	Produkt5	550.0	10	Tue Mar 28 02:00:00 CE...	495.0
6	Produkt6	660.0	10	Tue Mar 28 02:00:00 CE...	594.0
7	Produkt7	770.0	10	Tue Mar 28 02:00:00 CE...	693.0

Produkty

[1, Produkt1, 120.0, 10, Tue Mar 28 02:00:00 CEST 2017, 108.0]

[1, Produkt1, 120.0, 10, Tue Mar 28 02:00:00 CEST 2017, 108.0]

[2, Produkt2, 220.0, 20, Tue Mar 28 02:00:00 CEST 2017, 176.0]

[3, Produkt3, 320.0, 25, Tue Mar 28 02:00:00 CEST 2017, 240.0]

[4, Produkt4, 440.0, 50, Tue Mar 28 02:00:00 CEST 2017, 220.0]

[5, Produkt5, 550.0, 10, Tue Mar 28 02:00:00 CEST 2017, 495.0]

[6, Produkt6, 660.0, 10, Tue Mar 28 02:00:00 CEST 2017, 594.0]

[7, Produkt7, 770.0, 10, Tue Mar 28 02:00:00 CEST 2017, 693.0]

A Menu Inne Menu

Id produktu	Nazwa	Cena	Promocja	Data	Cena brutto
2	Produkt2	220.0	20	Tue Mar 28 02:00:00 CE...	176.0
3	Produkt3	320.0	25	Tue Mar 28 02:00:00 CE...	240.0
4	Produkt4	440.0	50	Tue Mar 28 02:00:00 CE...	220.0
5	Produkt5	550.0	10	Tue Mar 28 02:00:00 CE...	495.0
6	Produkt6	660.0	10	Tue Mar 28 02:00:00 CE...	594.0
7	Produkt7	770.0	10	Tue Mar 28 02:00:00 CE...	693.0

Produkty

[1, Produkt1, 120.0, 10, Tue Mar 28 02:00:00 CEST 2017, 108.0]

[1, Produkt1, 120.0, 10, Tue Mar 28 02:00:00 CEST 2017, 108.0]

[2, Produkt2, 220.0, 20, Tue Mar 28 02:00:00 CEST 2017, 176.0]

[3, Produkt3, 320.0, 25, Tue Mar 28 02:00:00 CEST 2017, 240.0]

[4, Produkt4, 440.0, 50, Tue Mar 28 02:00:00 CEST 2017, 220.0]

[5, Produkt5, 550.0, 10, Tue Mar 28 02:00:00 CEST 2017, 495.0]

[6, Produkt6, 660.0, 10, Tue Mar 28 02:00:00 CEST 2017, 594.0]

[7, Produkt7, 770.0, 10, Tue Mar 28 02:00:00 CEST 2017, 693.0]

3. Należy wykonać projekt z wykorzystaniem dodatkowych atrybutów dodanych do klasy Produkt1 np. Producent, Data_przydatności itp. – podobnie jak podczas laboratoriów z Technologii internetowych.